Reihe Informationsmanagement im Engineering Karlsruhe

Oliver Lohse

Entwicklung einer Methode zum Einsatz von Reinforcement Learning für die dynamische Fertigungsdurchlaufsteuerung

Band 1 – 2023



Oliver Lohse

Entwicklung einer Methode zum Einsatz von Reinforcement Learning für die dynamische Fertigungsdurchlaufsteuerung

Reihe Informationsmanagement im Engineering Karlsruhe
Band 1 – 2023

Herausgeber
Karlsruher Institut für Technologie
Institut für Informationsmanagement im Ingenieurwesen (IMI)
o. Prof. Dr. Dr.-Ing. Dr. h.c. Jivka Ovtcharova

Eine Übersicht aller bisher in dieser Schriftenreihe
erschienenen Bände finden Sie am Ende des Buchs.

Entwicklung einer Methode zum Einsatz von Reinforcement Learning für die dynamische Fertigungsdurchlaufsteuerung

von
Oliver Lohse



Karlsruher Institut für Technologie
Institut für Informationsmanagement im Ingenieurwesen

Entwicklung einer Methode zum Einsatz
von Reinforcement Learning für die
dynamische Fertigungsdurchlaufsteuerung

Zur Erlangung des akademischen Grades eines Doktors der Ingenieurwissenschaften von der KIT-Fakultät für Maschinenbau des Karlsruher Instituts für Technologie (KIT) genehmigte Dissertation

von Oliver Lohse, M.Sc.

Tag der mündlichen Prüfung: 15. Dezember 2022
Hauptreferent: Prof. Dr. Dr.-Ing. Dr. h. c. Jivka Ovtcharova
Korreferent: Prof. Dr.-Ing. Steffen Ihlenfeldt

Impressum



Karlsruher Institut für Technologie (KIT)
KIT Scientific Publishing
Straße am Forum 2
D-76131 Karlsruhe

KIT Scientific Publishing is a registered trademark
of Karlsruhe Institute of Technology.
Reprint using the book cover is not allowed.

www.ksp.kit.edu





Print on Demand 2023 – Gedruckt auf FSC-zertifiziertem Papier

ISSN 1860-5990
ISBN 978-3-7315-1282-0
DOI 10.5445/KSP/1000156002

Zusammenfassung

Kürzer werdende Produktlebenszyklen und der Wunsch des Kunden nach individualisierten Produkten führen zu kleineren Losgrößen in der Fertigung. Um diesen neuen Anforderungen gerecht zu werden, müssen Fertigungsunternehmen ihre Fertigungsstrategie anpassen. Aufgrund der wachsenden Produktvarianz scheint daher ein Wandel der klassischen Linienfertigung zu einer sogenannten Matrixfertigung sinnvoll. Letztere bietet mehr Freiheitsgrade bei der Planung und Steuerung der Fertigung und ermöglicht so auch die effiziente Fertigung von Produkten mit kleinen Losgrößen.

Die Freiheitsgrade der Matrixfertigung führen jedoch bei der Planung und Steuerung dazu, dass herkömmliche Berechnungsmethoden für die Fertigungsterminierung bzw. Regeln für die Steuerung nur begrenzt anwendbar sind. Tritt während der Fertigung eine Störung bzw. eine Abweichung von der geplanten Terminierung auf, müssen Mitarbeiter den Fertigungsprozess manuell umsteuern. Eine Neuterminierung nimmt aufgrund der Komplexität des Optimierungsproblems zu viel Zeit in Anspruch. Die manuelle Umsteuerung führt jedoch zu keinem Produktionsoptimum, was zu einer geringeren Produktionsauslastung führt.

Ziel der vorliegenden Arbeit ist es daher, eine Methode zu entwickeln, die in der Lage ist, die Matrixfertigung im Fall einer Störung zeitnah neuzuterminieren. Dafür werden verschiedene Methoden künstlicher Intelligenz auf neuartige Weise kombiniert. Die Validierung der entwickelten Methode erfolgt sowohl am Beispiel eines theoretischen als auch am Beispiel eines realen Terminierungsfalls.

Abstract

The customer's desire for individualized products and shorter product life cycles has led to smaller batch sizes in manufacturing. To meet these new requirements, manufacturing companies must adapt their production strategy. Due to the growing variation among products, a change from classic line production to so-called matrix production appears to be rational. Matrix production allows many more degrees of freedom in planning and controlling production and thus also enables the efficient manufacturing of products with small batch sizes.

However, the many degrees of freedom in matrix production mean that conventional calculation methods for production scheduling or control rules can only be used to a limited extent in planning and control. If a disruption or deviation from the planned scheduling occurs during production, employees must manually reschedule the production process. Rescheduling would take too much time due to the complexity of the optimization problem. However, manual rescheduling does not lead to an optimzed production, resulting in lower production utilization.

This thesis aims to develop a method that can reschedule the matrix production in the case of a disruption. For this purpose, different artificial intelligence methods are combined in a novel way. The developed method is validated on a theoretical and a real scheduling case.

Danksagung

Die vorliegende Arbeit ist während meiner Zeit als Doktorand in der Konzernforschung bei der Siemens AG entstanden. Der Austausch mit den Kollegen meiner Forschungsgruppe und den Kollegen aus verschiedenen Siemenswerken hat maßgeblich zu dieser Arbeit beigetragen, wofür ich mich an dieser Stelle bedanken möchte.

Mein besonderer Dank gilt Herrn Dr. -Ing Peter Robl, Leiter der Forschungsgruppe, für das mir entgegengebrachte Vertrauen und die Möglichkeit, eigenbestimmt arbeiten zu können. In zahlreichen Diskussionen über die Umsetzbarkeit der entwickelten Methode habe ich durch seine Praxiserfahrung viel lernen können.

Auch bedanke ich mich bei Frau Prof. Dr. Dr.-Ing. Dr. h. c. Jivka Ovtcharova für ihr Interesse und die Übernahme meiner Betreuung.

Des Weiteren danke ich Herrn Prof. Dr. -Ing. Steffen Ihlenfeldt für die Übernahme der Zweitbetreuung.

Nicht zuletzt gilt mein Dank den Studierenden, die mich während meiner Promotion in Form von Abschlussarbeiten, Praktika oder Werkstudententätigkeiten unterstützt haben.

Abschließend möchte ich meiner Mutter und meiner Großmutter für die Unterstützung während meiner Schul- und Studienzeit danken.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

APS	Advanced Planning and Scheduling-System
AVO	Arbeitsvorgang
BFS	Breitensuche
DFS	Tiefensuche
DQN	Deep-Q-Network
ERP	Enterprise-Resource-Planning
FAUF	Fertigungsauftrag
FIFO	First-in-First-out
FJSSP	Flexible-Job-Shop-Scheduling-Problem
GNN	Graph-Neural-Network
ID	Identifier
JSSP	Job-Shop-Scheduling-Problem
KI	Künstliche Intelligenz
ML	Machine Learning
MES	Manufacturing-Execution-System
MDP	Markov-Decision-Process
MDT	Mean-Down-Time

MTBF Mean-Time-Between-Failures

MCTS Monte-Carlo-Tree-Search

MADQN Multi-Agent-Deep-Q-Network

MAPPO Multi-Agent-Proximal-Policy-Optimization

NN Neuronales Netz

OSSP Open-Shop-Scheduling-Problem

PUCT Polynomial-Upper-Confidence-Trees

PPS Produktionsplanung und -steuerung

PPO Proximal-Policy-Optimization

SPT Shortest-Processing-Time

TLT Total-Lead-Time

UCB Upper-Confidence-Bound

UCT Upper-Confidence-Bound-for-Trees

1 Einleitung

In der vorliegenden Arbeit soll eine Methode zum Einsatz von Reinforcement Learning für die dynamische Fertigungsdurchlaufsteuerung entwickelt werden. Dieses einleitende Kapitel beinhaltet Erläuterungen zur Motivation, Problemstellung, Zielsetzung und zum Aufbau der Untersuchung.

1.1 Motivation

Fertigungsbetriebe müssen sich Herausforderungen wie einer steigenden Produktvarianz bis hin zur ‚Losgröße 1' und einer zunehmenden Anzahl globaler Mitbewerber stellen, um langfristig bestehen zu können [133]. Die wachsende Produktvarianz führt in einigen Bereichen zu einem Wandel der Fertigungsstrategie: weg von der klassischen Linienfertigung, hin zu einer zellorientierten, flexiblen Fertigung, ohne feste Taktzeit [58]. Die Terminierung dieser flexiblen Fertigung stellt ein komplexes Optimierungsproblem dar, bei dem verschiedene Produkte, Prioritäten und Bearbeitungszeiten so kombiniert werden müssen, dass eine zeit- und kostenoptimierte Fertigung ermöglicht wird [23] [137]. ‚Advanced-Planning-and-Scheduling'-Systeme, kurz APS, lösen das Optimierungsproblem mit unterschiedlichen (Meta-)Heuristiken und exakten Lösungsmethoden [17]. Meta-Heuristiken finden eine nahezu optimale Lösung für das Optimierungsproblem, benötigen dafür aber eine Berechnungszeit, die, abhängig von der Komplexität der Fertigung, Minuten bis Stunden betragen kann. Kommt es zu einer Störung in der Fertigung, werden die Produkte von einem verantwortlichen Mitarbeiter vor Ort umpriorisiert und umgesteuert, da eine Neuterminierung zu viel Zeit in Anspruch nehmen würde. Diese Umpriorisierung und Umsteuerung der Produkte geschieht

auf Basis der Erfahrung des Mitarbeiters und folgt keinen expliziten Regeln [81]. Im besten Fall wird für den betroffenen Fertigungsbereich ein lokales Optimum erreicht. Die gesamte Fertigung kann von dem Mitarbeiter nicht überblickt werden und wird dementsprechend auch nicht bei der lokalen Umpriorisierung und Umsteuerung betrachtet bzw. optimiert.

Methoden der ‚künstlichen Intelligenz' (KI) bieten neuartige Ansätze, mit deren Hilfe Unternehmen künftigen Herausforderungen begegnen können. Mit Beginn des 21. Jahrhunderts hat KI dank der sinkenden Kosten von Rechenleistung zunehmend an Bedeutung gewonnen. KI-Algorithmen werden seitdem vermehrt industriell eingesetzt [54]. Insbesondere in der Fertigung und Montage existieren noch viele unerschlossene Anwendungsfälle und Potenziale für Anwendungen, die auf Methoden der künstlichen Intelligenz beruhen [35]. Diese Methoden haben bereits gezeigt, dass sie sequenzielle Entscheidungsprobleme mit hoher Effizienz lösen können. So erlernen diese Algorithmen Spiele wie Go oder Schach und erreichen dabei eine übermenschliche Leistungsfähigkeit [121]. Aus technologischer Sicht erscheint es daher realistisch, dass KI-Algorithmen die Anforderungen der Echtzeit-Produktionssteuerung erfüllen können. Ein Vergleich verschiedener KI-Methoden im Hinblick auf deren Anwendung in der Fertigungssteuerung ist aufgrund unterschiedlicher Funktionsweisen und Implementationsmöglichkeiten noch ausstehend.

1.2 Problemstellung

Grundlage für den Fertigungsablauf ist das Terminierungsergebnis z. B. eines 'Advanced Planning and Scheduling'-Systems. Dieses IT-System fasst Fertigungsaufträge zusammen, priorisiert Produkte und ordnet den Fertigungsaufträgen verschiedene Fertigungsressourcen zu, unter anderem Menschen, Maschinen oder Werkzeuge. Das APS-System löst beim Erstellen der Terminierung ein Optimierungsproblem, das umso komplexer wird, je mehr Randbedingungen betrachtet werden müssen [23]. Randbedingungen sind z. B. die Anzahl der Maschinen und

Produkte, oder verschiedene Prioritäten der Fertigungsaufträge. Um das Optimierungsproblem zu lösen, werden z. B. (Meta-)Heuristiken eingesetzt, die eine näherungsweise Lösung finden. Wegen der Komplexität des Problems benötigen diese Heuristiken Minuten bis hin zu Stunden, um eine optimale Terminierung zu ermitteln [17] [23]. Tritt im Fertigungsprozess eine Störung auf, sind die Mitarbeiter daher gezwungen, die Produkte auf Grundlage ihrer Erfahrung umzupriorisieren und umzusteuern [3]. Dieser manuelle Eingriff in den Fertigungsprozess führt im besten Fall zu einem lokalen Optimum im entsprechenden Fertigungsbereich – welchen Einfluss die Umsteuerung der Produkte auf die nachfolgenden Fertigungsprozessschritte hat, ist jedoch für die Mitarbeiter in dem Moment nicht absehbar [41]. Das hat zur Folge, dass durch die manuellen Eingriffe kein globales Produktionsmaximum realisiert wird, was einen geringen Produktionsdurchsatz, eine Erhöhung der Durchlaufzeit und in letzter Konsequenz höhere Produktionskosten verursacht [41].

Die Anwendbarkeit von KI-Methoden zur Lösung einfacher Terminierungsprobleme wird seit einigen Jahren erfolgreich erforscht [82] [13] [7]. Dabei zeigt sich, dass unterschiedliche KI-Methoden dazu in der Lage sind, ein Terminierungsproblem nahezu optimal zu lösen. Die dabei untersuchten Terminierungsszenarien beschränken sich auf wissenschaftliche Instanzen ohne alternative Bearbeitungsmaschinen oder Arbeitsplätze. Häufig werden auch andere Randbedingungen wie Transport- und Rüstzeiten außer Acht gelassen [2]. Die Komplexität des Terminierungsproblems wird damit stark verringert, was dazu führt, dass sich diese Methoden nicht für den Einsatz in einer realen, dynamischen Fertigung eignen. Hinzu kommt, dass die veröffentlichten Terminierungsbenchmarks (z. B. in [34]), in der Regel das beste Ergebnis aus vielen und langen Trainings- und Validierungsläufen darstellen, da die Güte der Ergebnisse der einzelnen Läufe stark schwankt [132]. Das Terminierungsergebnis für eine reale Fertigung muss jedoch robust sein; d. h., das Ergebnis muss mit einer hohen Wahrscheinlichkeit zu einem globalen Produktionsmaximum beitragen. Eine KI-Methode, die diese Anforderungen erfüllt, wurde nach derzeitigem Kenntnisstand noch nicht publiziert.

Damit die zu entwickelnde KI-Methode das Terminierungsproblem im Falle einer Störung ausreichend gut lösen kann, müssen verschiedene Störungsszenarien Bestandteil des Trainings der KI sein. Mit einer steigenden Anzahl an Bearbeitungsmaschinen/Arbeitsplätzen und Produkten wächst auch die Zahl der potenziellen Störungsszenarien exponentiell an. Da aufgrund der Anzahl der Störungsszenarien nicht alle Störungen beim Training der KI-Methode betrachtet werden können, wird eine Generalisierungsfähigkeit der KI-Methode vorausgesetzt. Unterschiedliche Anwendungen z. B. in der Robotik zeigen jedoch, dass die Generalisierungsfähigkeit von KI-Methoden mitunter beschränkt ist [69] [30]. Die fehlende Generalisierungsfähigkeit stellt infolgedessen ein Hindernis für die Anwendung von KI-Algorithmen in der flexiblen Fertigung dar und muss daher besonders betrachtet werden.

1.3 Zielsetzung

In diesem Abschnitt werden die aus der obigen Problemstellung abgeleiteten Forschungsziele dargelegt. Dafür werden die entsprechenden Ziele formuliert, die im Verlauf der Untersuchung zu überprüfen sind.

Ziel dieser Arbeit ist es, eine Methode zu entwickeln, die im Falle einer Störung das Produktionsoptimum oder eine Lösung in der Nähe des Optimums herbeiführt. Das Produktionsoptimum wird hier als maximal möglicher Produktdurchsatz trotz Störung definiert. Dafür müssen bei der Umpriorisierung und Umsteuerung der Produkte alle Arbeitspläne inklusive alternativer Arbeitsvorgänge, Prioritäten und Maschinenzustände berücksichtigt werden.

Ziel 1: Mit Hilfe von KI-Methoden kann eine zeitnahe Terminierung der Fertigungsaufträge umgesetzt werden, die bei der Umpriorisierung/Umsteuerung von Produkten ein globales Produktionsoptimum bzw. eine Lösung nahe am Optimum herstellt.

In der vorliegenden Arbeit soll eine KI-Methode entwickelt werden, die in der Lage ist, die Fertigungsterminierung im Störungsfall dynamisch anzupassen. Dabei müssen zur Lösung des Terminierungsproblems sowohl alternative Maschinen/Arbeitsplätze und Arbeitsvorgänge als auch Transport- und Rüstzeiten betrachtet werden. Das Ergebnis der Terminierung muss zeitnah zur Verfügung stehen und mit einer hohen Wahrscheinlichkeit einen nahezu optimalen Produktionsablauf ermöglichen.

Ziel 2: KI-Methoden sind in der Lage auch komplexe Terminierungsprobleme mit vielen Freiheitsgraden nahezu optimal zu lösen. Zusätzlich kann eine Robustheit der Terminierungsergebnisse sichergestellt werden, die für Anwendungen in einer realen Fertigung ausreichend ist.

Die entwickelte KI-Methode muss unbekannte Störungsszenarien sicher lösen können. Dabei ist es notwendig, die vielfältigen Kombinationen aus Störungsort, -dauer und -zeitpunkt möglichst optimal zu verarbeiten. Dafür wird eine Trainingsmethode konzipiert, die die Generalisierungsfähigkeit von KI-Methoden steigert und infolgedessen deren Einsatz für die flexible Fertigungsterminierung ermöglicht.

Ziel 3: Mit Hilfe einer gezielten Trainingsmethode kann die Generalisierungsfähigkeit von KI-Methoden verbessert werden, sodass diese alle möglichen Störungsszenarien einer flexiblen Fertigung ausreichend lösen können.

1.4 Aufbau der Arbeit

Die vorliegende Arbeit gliedert sich in sechs Kapitel, deren Abfolge und Inhalt nachfolgend vorgestellt werden. Die grundsätzliche Gliederung lässt sich Abbildung 1.1 entnehmen.

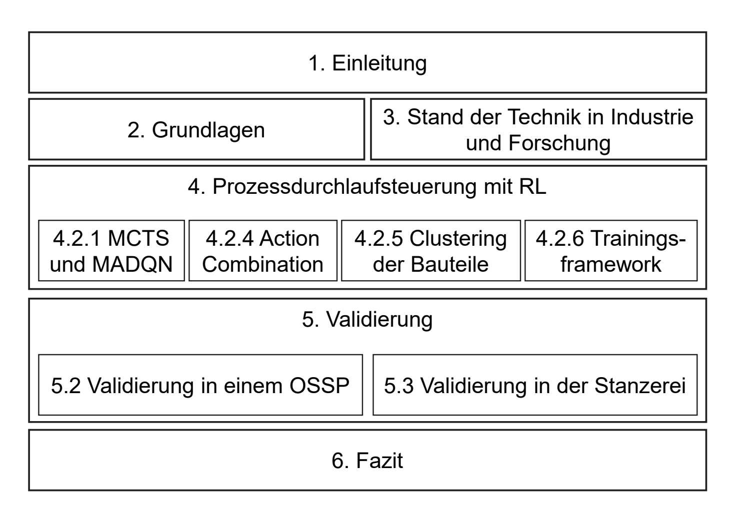


Abbildung 1.1: Aufbau der Arbeit

Die beiden Kapitel 2 und 3 *(Grundlagen und Stand der Technik in Industrie und Forschung)* bilden die Basis für die in der Arbeit entwickelten Methode. Im Kapitel 2 *Grundlagen* werden die Grundlagen der Produktionsplanung und -steuerung definiert; zusätzlich wird die Klassifizierung unterschiedlicher Terminierungsprobleme hergeleitet. Darüber hinaus werden für die spätere Methode relevante Ansätze der künstlichen Intelligenz und des Clusterns vorgestellt. Das Kapitel 3 *(Stand der Technik in Industrie und Forschung)* widmet sich den IT-Systemen, die in der Industrie für die Terminierung angewendet werden, sowie zusätzlich den verschiedenen Methoden, um Terminierungsprobleme zu lösen. Den Abschluss dieses Kapitels bildet die Feststellung der Forschungslücke, die diese Arbeit zu schließen versucht. Das Kapitel 4 *(Methode zum Einsatz von Reinforcement Learning zur Prozessdurchlaufsteuerung eines Produktionssystems)* umfasst die entwickelte Methode für die dynamische Terminierung samt aller Untermethoden wie dem Clustern der Bauteile oder dem Trainingsframework. Im Kapitel 5 *(Validierung)* wird die entwickelte Methode innerhalb zwei unterschiedlicher Terminierungsszenarien validiert. Den Abschluss der Arbeit bildet das Kapitel 6 *(Zusammenfassung und Ausblick)*.

2 Grundlagen

Das vorliegende Kapitel definiert die Grundlagen für die, im Verlauf der Arbeit zu entwickelnde, Methode. Dafür wird in den Abschnitten 2.1 bis 2.3 die Fertigung und deren Planungs- und Steuerungsprozesse definiert. Die Abschnitte 2.4 bis 2.6 befassen sich mit den Grundlagen von KI-Methoden.

2.1 Klassifizierung von Fertigungssystemen

In diesem Kapitel werden zunächst die unterschiedlichen Fertigungsarten vorgestellt. Darauf aufbauend werden die Fertigungsarten den verschiedenen Fertigungskonzepten zugeordnet.

2.1.1 Fertigungsarten

Der Begriff *Produktion* lässt sich in Montage und Fertigung unterteilen. Bei der Montage wird ein Produkt aus verschiedenen Bauteilen zusammengebaut. Merkmal der Fertigung ist die kontinuierliche Bearbeitung eines Bauteils. Dabei wird z. B. eine Welle in mehreren Arbeitsvorgängen und an mehreren Maschinen bearbeitet. Die vorliegende Arbeit beschäftigt sich mit den Zusammenhängen einer Fertigung, daher wird in den nachfolgenden Kapiteln immer von einer Fertigung ausgegangen. [38]

Fertigungsarten werden nach Auflagenhöhe (Losgröße) und Wiederholhäufigkeit (Anzahl produzierter Lose im Jahr) differenziert [38]. Anhand dieser Kriterien kann zwischen Einzel-, Kleinserien-, Großserien- und Massenfertigung unterschieden werden (vgl. Abbildung 2.1). Eine eindeutige Abgrenzung der einzelnen Fertigungsarten existiert nicht, da sich die einzelnen Bereiche überschneiden. So können Bauteile gleicher Stückzahl in der Komplexität variieren und folglich auch zu unterschiedlichen Zeitverteilungen führen. [38]

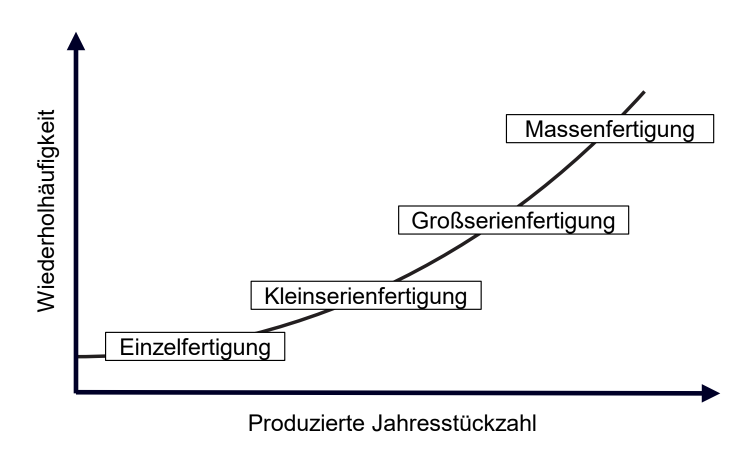


Abbildung 2.1: Einordnung der Fertigungsarten [38]

Die **Einzelfertigung** zeichnet sich durch häufige Auftragswechsel mit sehr kleinen Losgrößen aus. Der Anteil nicht wertschöpfender Tätigkeiten ist hier besonders groß, da ein ständiges Umrüsten der Maschinen notwendig ist. Zusätzlich unterscheiden sich die eingesetzten Werkzeuge und Maschinen in der Regel von denen der Massenfertigung, da der Spezialisierungsgrad in der Einzelfertigung geringer ist. Diese Art der Fertigung birgt ein großes Optimierungspotenzial, wenn der hohe Anteil nicht wertschöpfender Tätigkeiten verringert werden kann [39]. Ein Beispiel für die Einzelfertigung ist der Anlagenbau, bei dem jede Anlage spezifisch nach Kundenwunsch gefertigt wird. [38]

Die **Klein- und Großserienfertigung** liegt, bezogen auf die produzierte Jahresstückzahl und der Wiederholhäufigkeit, zwischen der Einzel- und der Massenfertigung. Typisch für die Serienfertigung ist eine Wiederholhäufigkeit von über zwölf Kundenaufträgen pro Jahr - das bedeutet, dass dieses Produkt mindestens einmal im Monat in einer gewissen Auflagenhöhe gefertigt wird. Der größere Automatisierungsgrad sowie der höhere Spezialisierungsgrad der Arbeitsvorgänge unterscheiden die Serienfertigung zusätzlich von der Einzelfertigung. [116]

Bei der **Massenfertigung** können Betriebsmittel speziell auf eine oder wenige Aufgaben ausgerichtet werden - der Spezialisierungsgrad steigt somit an. Rüstzeiten fallen aufgrund höherer Losgrößen seltener an, was einen höheren Auslastungsgrad der Maschinen zur Folge hat. Eine weitergehende Produktivitätssteigerung kann bei dieser Fertigungsart im Wesentlichen nur durch die Optimierung des Herstellungsprozesses herbeigeführt werden [39]. Von Massenfertigung wird z. B. üblicherweise bei der Herstellung von elektronischen Bauteilen gesprochen. [38]

2.1.2 Fertigungskonzepte

Das Fertigungskonzept hat einen großen Einfluss sowohl auf die Anordnung der Betriebsmittel, als auch auf die Organisationsstruktur der Fertigung. Neben der Anordnung der Betriebsmittel wird mit dem Fertigungskonzept zusätzlich der Materialfluss innerhalb der Fertigung festgelegt. Abbildung 2.2 stellt die fünf wichtigsten Fertigungskonzepte gegenüber. [38][50]

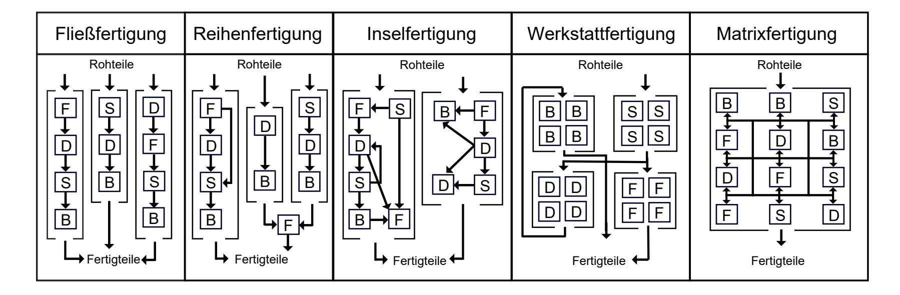


Abbildung 2.2: Fertigungskonzepte i. A. a. [38][50]

Bei der **Fließfertigung** folgt die räumliche Anordnung der Betriebsmittel dem Fertigungsablauf. Die Produkte durchlaufen dabei die Fertigung z. B. auf einem Fließband und werden sukzessive taktgesteuert an jedem Arbeitsplatz bearbeitet. Da die einzelnen Arbeitsplätze starr miteinander verbunden sind und zusätzlich einen hohen Spezialisierungsgrad aufweisen, wird dieses Fertigungskonzept in der Großserien- und der Massenfertigung angewendet. [134]

Ähnlich wie bei der Fließfertigung werden auch bei der **Reihenfertigung** die Arbeitsplätze dem Fertigungsverlauf folgend angeordnet. Der Materialfluss ist auch hier gerichtet, aber trotzdem können einzelne Arbeitsvorgänge übersprungen werden. Das Fertigungskonzept ist somit flexibler als die Fließfertigung und lässt unterschiedliche Ablaufvarianten der Produkte zu. [116]

Bei der **Inselfertigung** werden Arbeitsplätze mit verschiedenartigen Bearbeitungsverfahren zusammengefasst und bilden so weitestgehend autonome Fertigungseinheiten. Der Materialfluss innerhalb der Fertigungseinheiten ist ungerichtet und wird in der Regel von der Fertigungsinsel selbst gesteuert. [116]

Die **Werkstattfertigung** findet vor allem bei kleinen Losgrößen Anwendung. Dabei werden die Fertigungsarbeitsplätze nach ihren Bearbeitungsverfahren (Schweißen, Bohren, Fräsen etc.) zu räumlichen Einheiten zusammengefasst. Einen gerichteten Materialfluss wie bei der Fließfertigung gibt es bei diesem Fertigungskonzept nicht. [40]

Die **Matrixfertigung** besitzt den größten Freiheitsgrad aller Fertigungskonzepte. Die einzelnen Arbeitsplätze sind nicht dem Fertigungsverlauf folgend angeordnet, sondern ähneln im Grundlayout einer Matrix. Im Kontrast zur Fließfertigung kommt hier kein starres Fließband zum Einsatz, sondern ein flexibles Transportsystem, wie z. B. ein fahrerloses Transportsystem. So werden unterschiedliche Abarbeitungsreihenfolgen der Produkte ermöglicht. Dieses Fertigungskonzept schafft die Voraussetzungen für eine taktunabhängige Fertigung, die dazu in der Lage ist, eine große Varianz an Bauteilen zu fertigen. Die wesentliche Herausforderung in diesem Zusammenhang ist die Produktionsplanung und -steuerung. In der Arbeitsplanung müssen Arbeitspläne diesen hohen Grad an Flexibilität abbilden können (vgl. Abschnitt 2.2.3). Die Fertigungssteuerung muss zudem dynamisch auf Störungen reagieren können und bei der Auswahl des Alternativ-Arbeitsplatzes u. a. Faktoren wie Transportzeit, Rüstzeit und Kosten betrachten und dabei den gesamten Fertigungsprozess optimieren. [50]

Die Matrixfertigung soll aufgrund ihres großen Freiheitsgrades und der damit einhergehenden Komplexität des Optimierungsproblems als Grundlage für die in dieser Arbeit entwickelten Methode der Fertigungssteuerung dienen. Eine Fertigungssteuerung, die dieses Optimierungsproblem schnell und zuverlässig lösen kann, ist auch dazu in der Lage, die Steuerung der anderen weniger komplexen, Fertigungskonzepte abzudecken.

2.2 Produktionsplanung und -steuerung (PPS)

In diesem Abschnitt werden die Aufgaben und Ziele der Produktionsplanung und -steuerung analysiert. Detailliert wird auf Planungstätigkeiten wie die Arbeitsplanerstellung eingegangen und anschließend die Abgrenzung der Produktionssteuerung von der Terminierung dargelegt.

2.2.1 Aufgaben und Ziele der PPS

„Die Produktionsplanung und -steuerung umfasst die räumliche, zeitliche und mengenmäßige Planung, Steuerung und Kontrolle des gesamten Geschehens im Produktionsbereich“ [8]. Ziele der Produktionsplanung und -steuerung (PPS) sind hohe Termintreue, hohe und gleichmäßige Kapazitätsauslastung, kurze Durchlaufzeit, hohe Flexibilität und geringe Lager- bzw. Produktionsbestände [116]. Dabei lassen sich PPS-Prozesse anhand ihrer zeitlichen Tragweite in lang-, mittel- und kurzfristige Planungsaktivitäten einteilen (vgl. Abbildung 2.3). Während des langfristigen Produktionsplanungsprozesses werden strategische Rahmenbedingungen geschaffen, unter denen ein Unternehmen auch in Zukunft erfolgreich produzieren kann. Dazu gehört beispielsweise die Auswahl neuer Produktionsstandorte. Die mittelfristige Planung hat einen Planungshorizont von Wochen und Monaten. Dabei werden u. a. saisonale Nachfrageschwankungen, Mitarbeiterbedarf und der Ressourcenbedarf der nächsten Monate bestimmt. Einzelne Fertigungsaufträge (FAUFs) (vgl. Abschnitt 2.2.2) werden hierbei nicht berücksichtigt. [53] [123]



Abbildung 2.3: Planungshorizonte der Produktionsplanung und -steuerung [123]

Entscheidungen in der kurzfristigen Produktionsplanung umfassen das operative Produktionsprogramm und die dafür benötigten Ressourcen sowie die zeitliche Abwicklung der Produktion [129]. Dabei werden Losgrößen festgelegt und Kundenaufträge zu FAUFs kombiniert. Diese Planungsphase erstreckt sich über einen Planungshorizont von Tagen und Wochen. [40] [116]

Sowohl die lang- als auch die mittelfristige Planung werden im Verlauf der vorliegenden Arbeit nicht weiter betrachtet, weswegen auf diese Bereiche hier nicht detaillierter eingegangen wird. Die kurzfristige Planung wird zur stärkeren Abgrenzung von den anderen Planungshorizonten auch als Terminierung bezeichnet [18]. Das Terminierungsergebnis enthält einen Produktionsplan der FAUFs, der die Maschinenbelegung und Ressourcenverfügbarkeit berücksichtigt. Bei Abweichung von der Planung, z. B. bei Produktionsstörungen oder Eilaufträgen, muss eine Neuplanung angestoßen werden. Die Neuplanung kann mehrere Stunden beanspruchen in denen kein ausführbarer bzw. ein veralteter Produktionsplan vorliegt [51]. Aus diesem Grund gibt es in der Fertigung Mitarbeiter die ‚Steuerer' genannt werden. Sie greifen im Störungsfall in den Produktionsablauf ein und steuern die FAUFs um. Eine Planungsgrundlage gibt es für diese Eingriffe nicht. Die Komplexität von Fertigungen mit vielen unterschiedlichen Produkten und alternativen Maschinen ist so hoch, dass Mitarbeiter nicht in der Lage sind, die Produktionsplanung ad-hoc zu übernehmen und dabei ein Produktionsoptimum herbeizuführen [57]. In der Regel steuert der Mitarbeiter die FAUFs auf Basis seines Erfahrungswissens oder mit Hilfe einfacher Regeln bzw. Heuristiken (vgl. Abschnitt 3.2.2) um. Diese manuellen Eingriffe haben Auswirkungen auf alle Fertigungsschritte und sind daher zu vermeiden. [3]

2.2.2 Aufbau und Inhalt des Arbeitsplanes

Der Arbeitsplan ist das Ergebnis der Arbeitsplanung und in der Regel termin- und auftragsneutral. Der Verband für Arbeitsgestaltung, Betriebsorganisation und Unternehmensentwicklung (REFA) definiert den Arbeitsplan wie folgt: „Darin ist die Vorgangsfolge zur Fertigung eines Teiles, einer Gruppe oder eines Erzeugnisses beschrieben; dabei sind mindestens das verwendete Material sowie für jeden Arbeitsvorgang der Arbeitsplatz, die Betriebsmittel, die Vorgabezeiten und gegebenenfalls die Lohngruppe angegeben" [109]. Der Arbeitsplan besteht aus einzelnen Arbeitsvorgängen (AVOs), die für die Fertigung und/oder Montage eines Produkts notwendig sind (vgl. Abbildung 2.4). Dem AVO sind mindestens der Arbeitsplatz, das zu verwendende Material und die Betriebsmittel zuzuordnen

[108]. Der Begriff *Arbeitsplan* dient dabei nur als Oberbegriff und kann weiter in Fertigungs- und Montagearbeitsplan unterteilt werden. Diese unterscheiden sich in Umfang und Aufbau.

<table>
<tr><td colspan="2" rowspan="2">Blatt 1 von 1</td><td colspan="2">Datum: 01.04.2019</td><td colspan="2" rowspan="2">Auftrags-Nr.: 3103-2022</td><td colspan="2" rowspan="2">Arbeitsplan</td></tr>
<tr><td colspan="2">Bearbeiter: P. Robl</td></tr>
<tr><td colspan="2">Stückzahl: 30</td><td colspan="2">Bereich: 1-20</td><td colspan="2">Benennung: Antriebswelle</td><td colspan="2">Zeichnungs-Nr.: 170-0542</td></tr>
<tr><td colspan="2">Werkstoff: St50</td><td colspan="2">Rohform und – abmessungen: Rundmaterial Ø 60mm</td><td colspan="2">Rohgewicht.: 7,6 kg</td><td colspan="2">Fertiggewicht.: 4,6 kg</td></tr>
<tr><td>AVO Nr.</td><td>Arbeitsvorgangsbeschreibung</td><td>Kostenstelle</td><td>Lohngruppe</td><td>Maschine</td><td>Fertigungshilfs-mittel</td><td>Rüstzeit [min]</td><td>Bearbeitungszeit [min]</td></tr>
<tr><td>10</td><td>Rundmaterial auf 345mm Länge sägen</td><td>300</td><td>04</td><td>4101</td><td>-</td><td>5</td><td>10</td></tr>
<tr><td>20</td><td>Rundmaterial auf 340mm ablängen und zentrieren</td><td>340</td><td>08</td><td>4201</td><td>1001
1051</td><td>2</td><td>15</td></tr>
<tr><td>30</td><td>Welle komplett drehen</td><td>360</td><td>08</td><td>4313</td><td>1101/1121/1131</td><td>2,6</td><td>20</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td><td>...</td></tr>
<tr><td>70</td><td>Fertigteilkontrolle</td><td>900</td><td>07</td><td>9002</td><td>-</td><td>3,8</td><td>10</td></tr>
</table>

Abbildung 2.4: Arbeitsplan mit Kopfdaten und Arbeitsvorgängen [40]

Der Fertigungsarbeitsplan wird z. B. zur Zerspanung von Rohteilen und bei der Fertigung von Einzelteilen genutzt. Anders als in der Montage gibt es bei der Einzelteilfertigung nur ein Rohmaterial, das in einem oder mehreren Arbeitsvorgängen bearbeitet wird. So wird beispielsweise aus einem zylindrischen Vollmaterial in mehreren Arbeitsvorgängen eine Welle mit verschiedenen Absätzen gefertigt. Sowohl der Fertigungs- als auch der Montagearbeitsplan ist auftrags- und terminneutral und wird erst nach Eingang und Annahme der Kundenbestellung in einen Kundenauftrag (KAUF) und anschließend, als Ergebnis der Terminierung, in einen Planauftrag sowie letztlich in einen Fertigungsauftrag (FAUF) mit Start- und Endterminen überführt (vgl. Abschnitt 2.2.4). [16]

2.2.3 Parallele und alternative Arbeitsfolgen innerhalb des Arbeitsplanes

Bei der Arbeitsplanerstellung ist die Flexibilität des Fertigungskonzeptes ausschlaggebend. Bei einer Fließfertigung (vgl. Abschnitt 2.1.2) sind die einzelnen Arbeitsplätze auf wenige Aufgaben spezialisiert und starr über ein Fließband miteinander verbunden. Bei Störungen können die Fertigungsaufträge in diesem Fertigungskonzept nicht umgesteuert werden - das hat zur Folge, dass im schlimmsten Fall das Fließband und damit die komplette Fertigung angehalten werden muss, weil eine Störung auftritt. [115]

Können Arbeitsvorgänge nicht nur auf der sogenannten Stammlinie, sondern auch z. B. auf parallelen Fertigungslinien durchgeführt werden, wird von **parallelen Folgen** gesprochen (vgl. Abbildung 2.5). **Alternative Folgen** werden genutzt, um bei kurzzeitigen Kapazitätsengpässen auf andere Arbeitsvorgänge auszuweichen. [16]

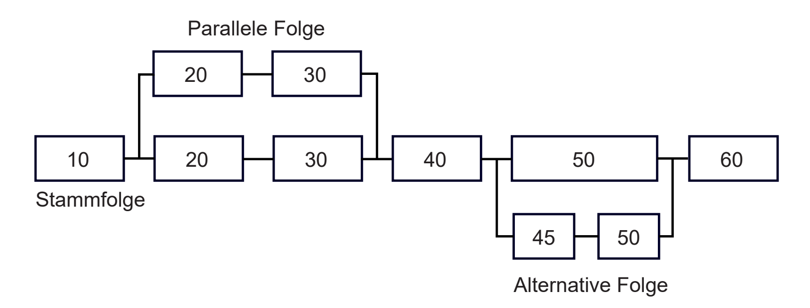


Abbildung 2.5: Parallele und alternative Folgen des Arbeitsplanes [16]

Das Matrix-Fertigungskonzept bietet, wie in Abschnitt 2.1.2 erwähnt, einen sehr großen Freiheitsgrad in der Fertigung. Produkte können frei zwischen den Arbeitsplätzen transportiert werden und unterliegen so nur bedingt den Auswirkungen von Störungen. Die hohe Flexibilität der Fertigung setzt aber auch ein Umdenken in der Arbeitsplanung voraus. Um die Vorteile der Matrixfertigung nutzen zu können müssen die Arbeitspläne erweitert werden. Parallele und alternative Folgen

bilden die Grundlage für diese Flexibilität. Die unterschiedlichen Arbeitsplätze können z. B. in Bearbeitungsgeschwindigkeit, Kosten oder Automatisierungsgrad voneinander abweichen. Zusätzlich müssen aber auch Freiheitsgrade bei der Fertigungsreihenfolge betrachtet werden - dafür sind Vor- und Nachbedingungen der einzelnen Arbeitsvorgänge festzulegen (vgl. Abbildung 2.6). Insbesondere bei einer nicht taktgesteuerten Fertigung kann so deren Auslastung konstant gehalten werden. Neben der Optimierung der Fertigungsauslastung ermöglichen die verschiedenen Ablaufpfade auch eine Optimierung hinsichtlich der Durchlaufzeit, der Fertigungskosten und des Energieverbrauches. [40]

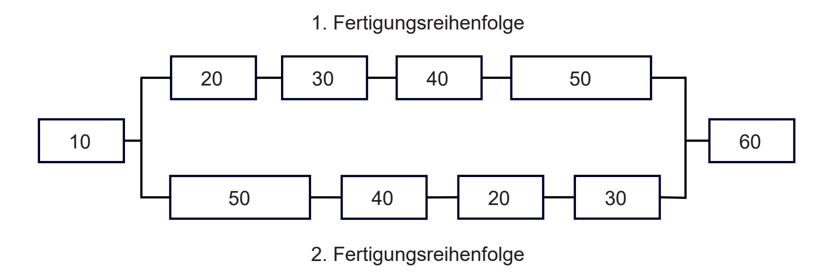


Abbildung 2.6: Vergleich unterschiedlicher Fertigungsreihenfolgen

2.2.4 Auftragsterminierung

Der in den Abschnitten 2.2.2 und 2.2.3 beschriebene Arbeitsplan bildet die Grundlage für die Auftragsterminierung. Bei der Auftragsterminierung wird die priorisierte Ablaufvariante des Arbeitsplans ermittelt und mit aktuellen und prognostizierten Informationen erweitert. Ziel der Auftragsterminierung ist die zeitliche Festlegung jedes Arbeitsvorganges. Dafür müssen u. a. die aktuelle und die prognostizierte Auftragslage, die Schichtpläne sowie die vorhandenen Maschinen- und Personalkapazitäten berücksichtigt werden. [73]

Die Terminierung der einzelnen Arbeitsschritte kann mittels Vorwärts- oder Rückwärtsterminierung erfolgen. Bei Ersterer wird der frühestmögliche Termin für die Fertigung zugrunde gelegt und mit Hilfe des Arbeitsplans der frühestmögliche Beginn des jeweiligen Arbeitsvorganges terminiert. Bei der Rückwärtsterminierung wird der mit der Annahme des Kundenauftrages verbindlich zugesagte Liefertermin als Grundlage genommen. Daraus können dann anhand des Arbeitsplans und der darin enthaltenen Fertigungszeiten Start- und Endtermine der einzelnen Arbeitsvorgänge berechnet werden. Voraussetzung hierfür ist, dass sämtliche AVOs mit den jeweiligen Fertigungszeiten, die für die Terminierung relevant sind, im Arbeitsplan enthalten sind. [16]

Das Ergebnis einer erfolgreichen Auftragsterminierung ist ein terminierter Fertigungsauftrag (FAUF). Dieser FAUF enthält die einzelnen AVOs des Arbeitsplans und zusätzlich deren terminierte Anfangs- und Endzeit. Kommt es in der Fertigung oder Montage zu Störungen oder muss ein nicht terminierter Eilauftrag ausgeführt werden, sind vorher terminierte FAUFs ausgesetzt. In dieser Situation entscheidet der Meister oder ein Steuerer spontan, wie Arbeitsschritte und Terminierung angepasst werden sollen oder ob eine Neuterminierung angestoßen wird (vgl. Abschnitt 2.2.1). Wird ohne Neuterminierung auf die Störung reagiert, hat die Produktionsleitebene keine Informationen darüber, welche FAUFs aktuell in welcher Reihenfolge produziert werden. Erst über die Rückmeldung der abgeschlossenen AVOs oder FAUFs kann nachverfolgt werden, welche Fertigungsschritte abgeschlossen sind. Informationen über den aktuellen Stand der Fertigung werden aber zum einen für die Terminierung zukünftiger FAUFs benötigt und bilden zum anderen die Grundlage für einen ‚virtuellen Zwilling' des Produktes und der Fertigung. [16]

Terminierungs- und Neuterminierungs-Ansätze lassen sich in vier verschiedene Strategien unterteilen:

1. Die **vollständig prädiktive Terminierung** beschreibt eine Terminierung mit Start- und Endterminen für die FAUFs. Störungen und andere unvorhergesehene Ereignisse werden dabei nicht betrachtet - die Fertigung wird als deterministisch angenommen. [55]

2. Von **robust-proaktiver Terminierung** wird gesprochen, wenn vor der Fertigung eine Terminierung erstellt wird, die im Gegensatz zur vollständig, prädiktiven Terminierung auch Störungen berücksichtigt. Um eine robuste Terminierung erzeugen zu können, müssen im Vorfeld Informationen über mögliche Störungen und deren Auswirkungen vorliegen. Maschinenstörungen können wie in Abschnitt 2.2.5 geschildert, mittels einer Störungswahrscheinlichkeit und einer durchschnittlichen Störungsdauer beschrieben werden. Störungen die z. B. aufgrund von fehlendem Material auftreten, können nicht über Durchschnittswerte abgebildet werden und finden daher bei der robust-proaktiven Terminierung auch keine Berücksichtigung. [55]

3. Bei der **prädiktiv-reaktiven Terminierung** wird vor der Fertigung eine Terminierung durchgeführt, analog zur vollständig prädiktiven Terminierung. Treten während der Fertigung Störungen auf, wird die zuvor erstellte Terminierung angepasst. Die Neuterminierung kann mittels einer sogenannten Terminierungs-Reparatur oder einer Neuterminierung erfolgen. Die **Terminierungs-Reparatur** hat dabei das Ziel, die bestehende Terminierung der FAUFs größtenteils beizubehalten und kann in zwei Ansätze aufgeteilt werden. Die *‚partielle Neuterminierung‘* verfolgt den Zweck, so schnell wie möglich zur ursprünglich geplanten Terminierung zurückzukehren. Dafür werden nur die von einer Störung betroffenen FAUFs neu terminiert. Der zweite Ansatz ist das sogenannte *‚Right-Shift-Scheduling‘*. Dabei werden alle FAUFs so lang in die Zukunft verschoben, bis eine Terminierung durchführbar ist. Beide Ansätze benötigen weniger Berechnungsressourcen als die Neuterminierung und bewahren die Stabilität des Fertigungssystems. Bei der **Neuterminierung** wird die zuvor geplante Terminierung verworfen und eine komplett neue erstellt. Die Neuterminierung kostet zum einen Zeit und zum anderen kann es zu Instabilitäten in der Fertigung kommen, wenn bei jeder Störung neu terminiert wird. Ziel ist es daher, eine Neuterminierung nur in absoluten Ausnahmefällen durchzuführen. [33] [101]

4. Bei der **reaktiven Terminierung** wird vor der Fertigung keine Terminierung erzeugt. Alle Arbeitsabwicklungs- und Entscheidungsprozesse werden lokal und ad-hoc verwirklicht. Die Grundlage für die Arbeitsabwicklungs- und Entscheidungsprozesse bildet die aktuelle Situation in der Fertigung. Einfache Heuristiken (vgl. Abschnitt 3.2.2) wie First-in-First-out, Earliest-Due-Date oder eine Sortierung nach der FAUF-Priorität werden bei der reaktiven Terminierung genutzt, um den zu bearbeitenden FAUF aus einer Liste anstehender Fertigungsaufträge auszuwählen. Logistik-Regeln sind schnell und einfach zu implementieren, führen aber nur in den wenigsten Fällen zu einer optimalen Auslastung der Fertigungskapazitäten [140]. Zusätzlich lässt sich bei dieser Terminierungsvariante im Vorfeld nur schwer abschätzen, wie viele FAUFs gefertigt werden können, bzw. ob die Fertigungskapazitäten und andere Ressourcen in geeigneter Menge zur Verfügung stehen. [101]

2.2.5 Störungsmanagement

Störungen entstehen immer dann, wenn Plan-Zeiten oder -Mengen nicht mit den aktuellen Ist-Zeiten oder -Mengen übereinstimmen. Allgemein werden Störungen als „[...] Einflussfaktoren bezeichnet, die von außen (exogen), aber auch vom Prozess selbst (endogen) unabhängig von anderen Größen, und damit mehr oder weniger zufällig, auf einen Prozess einwirken, sodass eine in der Regel quantifizierbare Abweichung des tatsächlich erzielten Prozessergebnisses (Ist-Größe) von einer a priori definierten Soll-Größe entsteht.“ [122]

Eine Unterteilung kann in Ressourcen-bezogene und FAUF-bezogene Störungen erfolgen (vgl. Tabelle 2.1). Als **Ressourcen-bezogene Störungen** sind Störungen definiert, die aufgrund fehlender Materialien und Werkzeuge oder einer Bearbeitungsmaschinenstörung auftreten. Als **FAUF-bezogene Störungen** werden Störungen klassifiziert, die im Zusammenhang mit einem FAUF stehen, wenn z. B. ein FAUF storniert wird oder sich dessen Losgröße oder Priorität ändert. [32][125]

Tabelle 2.1: Klassifizierung von Störungen in Ressourcen- und FAUF-bezogen [32][125]

Ressourcen-bezogen	FAUF-bezogen
Bearbeitungsmaschinenstörung	ungeplante Eilaufträge
Werkzeug ist nicht verfügbar	Änderung der FAUF-Priorität
Probleme bei der Materialversorgung	Stornierung von FAUFs
fehlender Maschinenbediener	Änderung des Fertigstellungsdatums
Stromausfall	Losgrößen-Anpassung

Ressourcen-bezogene Störungen haben einen direkten Einfluss auf die Verfügbarkeit der Bearbeitungsmaschine. Prinzipiell kann eine Bearbeitungsmaschine zwei sich abwechselnde Betriebszustände annehmen: ‚in Betrieb‘ und ‚außer Betrieb/gestört‘. Die Verfügbarkeit ergibt sich aus dem Verhältnis zwischen dem Betriebszustand ‚in Betrieb‘ und der Gesamtzeit, z. B. einer Schicht, und ist Teil der Berechnung für die Kennzahl ‚Overall Equipment Effectiveness‘ (OEE). Die Betriebszeit wird als MTBF (en: Meantime Between Failures) bezeichnet. Die Gesamtzeit setzt sich aus der MTBF und der Zeit, in der die Bearbeitungsmaschine außer Betrieb ist, der MDT (en: Mean Down Time), zusammen. Die Verfügbarkeit einer Bearbeitungsmaschine K berechnet sich somit wie in Gleichung 2.1 dargestellt. [37]

$$K = \frac{MTBF}{MTBF + MDT} \tag{2.1}$$

Des Weiteren lassen sich stochastische und deterministische Störungen differenzieren [18]. Stochastische Störungen sind Störungen, wie sie in Tabelle 2.1 beschrieben sind. Diese Störungen können nicht vorhergesagt werden und unterliegen einer Wahrscheinlichkeitsverteilung. Deterministische Störungen treten auf, wenn z. B. eine geplante Instandhaltungsmaßnahme an einer Bearbeitungsmaschine vorgenommen wird. Im Rahmen dieser Arbeit werden ausschließlich

stochastische Störungen behandelt. Da deterministische Störungen eingeplant werden können, stellen sie auch keine Störung im eigentlichen Sinne der Störungsdefinition dar. [18]

2.3 Klassifikation von Terminierungsproblemen

Nachfolgend werden unterschiedliche mathematische Beschreibungen von Terminierungsproblemen vorgestellt. Zusätzlich wird auf verschiedene Optimierungsziele bei der Lösung der Terminierungsprobleme eingegangen.

2.3.1 Stationskonfiguration

Die Stationskonfiguration definiert die funktionellen Anordnungen der einzelnen Arbeitsplätze bzw. Stationen in der Fertigung und beschreibt somit die in Abschnitt 2.1.2 vorgestellten Fertigungskonzepte mathematisch als Terminierungsproblem. Diese Terminierungsprobleme müssen gelöst werden, um eine optimale Terminierung der Fertigung zu gewährleisten. In Abbildung 2.7 sind die wesentlichen Ausprägungen der Stationskonfigurationen dargestellt, aus denen sich Terminierungsprobleme ableiten lassen. [106]

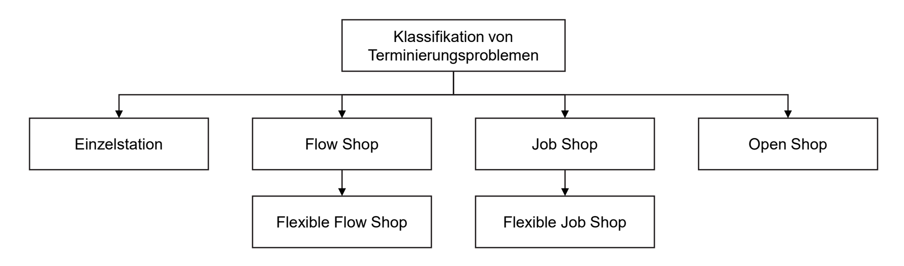


Abbildung 2.7: Klassifikation von Terminierungsproblemen

Das **Einzelstation-Terminierungsproblem** bezeichnet ein Terminierungsproblem, in dem nur ein beplanbarer Arbeitsplatz vorhanden ist. Diese Ausprägung des Terminierungproblems stellt einen leicht zu lösenden Sonderfall dar. [106]

Das **Flow-Shop-Scheduling-Problem** beschreibt das Terminierungsproblem einer Fließfertigung (vgl. Abschnitt 2.2). Dabei durchlaufen die FAUFs ein starres Fertigungssystem, das keine Bearbeitungsalternativen enthält. Alle FAUFs durchlaufen die Arbeitsplätze in der selben Reihenfolge (vgl. Abbildung 2.2). Zusätzlich können sich FAUFs nicht überholen und werden nach dem ‚First-in-First-Out' Prinzip abgearbeitet. [106]

Eine spezielle Form des Flow-Shop-Scheduling-Problems ist das **Flexible-Flow-Shop-Scheduling-Problem**. Auch hier durchlaufen die FAUFs alle Arbeitsplätze in der selben Reihenfolge, aber es gibt für jeden Arbeitsplatz mehrere Alternativen. Das Flexible-Flow-Shop-Problem kann sowohl bei der Reihenfertigung als auch in der Gruppenfertigung (vgl. Abschnitt 2.1.2) angewendet werden. [42] [106]

Beim **Job-Shop-Scheduling-Problem** (JSSP) folgt jeder FAUF seinem eigenen Arbeitsplan (vgl. Abschnitt 2.4) durch die Fertigung. Die Arbeitspläne der einzelnen FAUFs können sich dabei unterscheiden. Ähnlich wie beim Flow-Shop-Scheduling-Problem gibt es auch hier keine alternativen Bearbeitungsplätze. Mathematisch kann der Lösungsraum dieses Problems folgendermaßen beschrieben werden [106]: m Bearbeitungsmaschinen $(M_1, M_2, ..., M_n)$ sollen j FAUFs $(J_1, J_2, ..., J_n)$ fertigen. Die Bearbeitungszeiten der einzelnen Bearbeitungsmaschinen können voneinander abweichen. Jeder FAUF j besitzt einen Arbeitsplan, der aus n Arbeitsvorgängen besteht, die in einer vorgeschriebenen Reihenfolge abgearbeitet werden müssen. Daraus ergeben sich $(j!)^m$ Möglichkeiten, die FAUFs zu fertigen [140]. Bei einer Fertigung mit zehn Maschinen und zehn FAUFs resultiert hieraus ein Lösungsraum mit $3.9 \cdot 10^{65}$ Lösungsmöglichkeiten. Das JSSP wird den NP-schweren Problemen zugeordnet und gehört damit zur Klasse der am schwierigsten zu lösenden mathematischen Probleme [75]. Dieses Terminierungs-Problem muss gelöst werden, wenn eine Terminierung für die Inselfertigung (vgl. Abschnitt 2.1.2) erstellt wird, bei der es keine alternativen Bearbeitungsplätze gibt. [135] [2]

Das **Flexible-Job-Shop-Scheduling-Problem** (FJSSP) erweitert das Job-Shop-Problem um alternative Arbeitsplätze. Dadurch ergeben sich je nach Anzahl der alternativen Arbeitsplätze noch mehr Möglichkeiten für den Produktionsablauf als beim JSSP. Auch dieses Terminierungsproblem gehört den NP-schweren Problemen an. Es ist zu lösen, wenn die Terminierung einer Matrixfertigung (vgl. Abbildung 2.2) erfolgen soll. Die für einen Teil der Validierung in Kapitel 5 dieser Arbeit gewählte Stanzerei entspricht einer Matrixfertigung, weshalb das FJSSP für die Validierung dieser Fertigung gelöst werden muss. [42] [106]

Das **Open-Shop-Scheduling-Problem** (OSSP) beschreibt das komplexeste Terminierungsproblem. Dabei wird, anders als beim Flexible-Job-Shop-Scheduling-Problem, keine Reihenfolge der Bearbeitungsschritte festgelegt. Alle FAUFs können alle Maschinen in jeder Reihenfolge durchlaufen. Die Anzahl der möglichen Fertigungsabläufe steigt im Vergleich zum Job-Shop-Scheduling-Problem und dem Flexible-Job-Shop-Scheduling-Problem deutlich an. Der Lösungsraum des OSSPs ergibt sich aus der Anzahl der Maschinen und FAUFs und berechnet sich folgendermaßen: $(m \cdot j)!$ [140]. Selbst bei kleinen Instanzen mit zehn Maschinen und zehn FAUFs resultieren daraus $9.33 \cdot 10^{157}$ mögliche Lösungen. Dieses Terminierungsproblem setzt Annahmen voraus, die in der Fertigung so nicht vorzufinden sind. So kann mit diesem Terminierungsproblem kein Arbeitsplan abgebildet werden, weil die Reihenfolge der Arbeitsschritte frei wählbar ist. Das OSSP wird aber trotzdem auch in den nachfolgenden Kapiteln betrachtet, da es das Terminierungsproblem mit der höchsten Komplexität beschreibt und grundsätzlich davon auszugehen ist, dass ein Algorithmus, der dieses Problem lösen kann, auch in der Lage ist, weniger komplexe Terminierungsprobleme, wie das FJSSP zu lösen. [72] [48] [106]

2.3.2 Zielsetzung der Optimierung für die Terminierung

Bei der Optimierung der Terminierung gibt es mehrere unterschiedliche Zielsetzungen. Im Folgenden werden ausgewählte Optimierungszielgrößen vorgestellt, die minimiert werden sollen.

Die **Maximum Lateness** (L_{max}) (de: maximale Verspätung) gibt die größte Differenz aller FAUFs zwischen Fertigstellungszeitpunkt C und Fälligkeitsdatum d an. Die Lateness L_j für FAUF j berechnet sich somit wie in Formel 2.2 dargelegt. Ziel ist es hier, die maximale Lateness zu minimieren, um z. B. Vertragsstrafen wegen einer verspäteten Produktauslieferung möglichst gering zu halten. [106]

$$L_j = C_j - d_j \tag{2.2}$$

Die **Makespan** (C_{max}) (de: Fertigungsdauer) gibt an, wie viel Zeit zur Fertigung aller FAUFs j benötigt wird, und ist äquivalent zur Fertigstellungszeit des letzten Bauteils (vgl. Formel 2.3). Die Minimierung der Makespan ist Sinnvoll, wenn z. B. lackierte Bauteile montiert werden sollen. Haben die einzelnen Bauteile unterschiedliche Trocknungszeiten, ist für die Montage des Produktes die Trocknungszeit desjenigen Bauteils, bei dem der Lack zuletzt getrocknet ist, besonders ausschlaggebend. Nach einer Makespan-Optimierung wird in diesem Fall das Produkt mit der längsten Trocknungszeit als erstes lackiert werden, um C_{max} zu minimieren. Ähnliches gilt z. B. beim Beladen eines LKWs, der erst losfahren kann, wenn alle Ladungsgüter eingeladen sind, wobei die Beladezeit jedes einzelnen Ladungsgutes keine Rolle spielt. Wird die Makespan C_{max} minimiert, kann in der Regel auf eine bessere Maschinenausnutzung geschlossen werden. [106]

$$C_{max} := max\{C_j | j \in J\} \tag{2.3}$$

Gibt es FAUFs mit unterschiedlichen Prioritäten, muss die Makespan je nach FAUF-Priorität gewichtet werden. Aus diesem Grund wird eine zusätzliche Gewichtung w eingeführt - dabei wird von einer **gewichteten Makespan** $C_{max,w}$ gesprochen. [87]

Da sich einzelne Ausreißer stark auf die Makespan C_{max} auswirken, kann auch die **Total Lead Time** TLT_j (de: gesamte Durchlaufzeit) als Optimierungsgröße herangezogen werden. Anders als bei der Makespan C_{max}, bei der nur der FAUF mit der längsten Produktionszeit ausschlaggebend ist, werden bei der Total Lead

Time TLT_j die einzelnen Fertigungszeiten der FAUFs summiert. Eine Verbesserung der Fertigungszeit eines einzelnen FAUFs hat damit Auswirkungen auf die Total Lead Time und lässt so einen genaueren Vergleich von Fertigungsdurchläufen zu. Auch die Total Lead Time kann gewichtet werden, indem die gewichteten Fertigstellungszeiten $C_{j,w}$ aller FAUFs aufsummiert werden (vgl. Formel 2.4). [106]

$$TLT_{j,w} = \sum_{j \in J} C(j, w_j) \tag{2.4}$$

Zur Validierung der zu entwickelnden Methode wird die Total Lead Time TLT_j bzw. die gewichtete Total Lead Time $TLT_{j,w}$ herangezogen (vgl. Formel 2.4). Zum einen lässt die Total Lead Time eine detailliertere Auswertung als die Makespan $C_{max,w}$ zu und zum anderen können damit FAUF-Prioritäten abgebildet werden. [106]

2.4 Maschinelles Lernen und neuronale Netze

Für den weiteren Verlauf der Arbeit spielen insbesondere neuronale Netze sowie die Lern-Methoden *Supervised*, *Unsupervised* und *Reinforcement Learning* eine zentrale Rolle, weswegen diese Methoden in den nachfolgenden Abschnitten vorgestellt werden.

2.4.1 Definition des Begriffs der künstlichen Intelligenz

Der Begriff der künstliche Intelligenz wird zum ersten Mal 1956 von John McCarthy definiert als „die Wissenschaft und Technik der Herstellung intelligenter Maschinen" [66]. Intelligenz ist in diesem Kontext als komplexe und menschenähnliche Informationsverarbeitung zu verstehen; da es sowohl in der Neurobiologie als auch in der Psychologie an einer einheitlichen Definition für den Intelligenzbegriff mangelt, ist er auch in der Informatik nicht vollständig abgrenzbar

[102]. Zur Unterscheidung und Bewertung, ob ein Computerprogramm intelligent ist oder nicht, können verschiedene Tests angewendet werden. Der erste und einer der bekanntesten ist der Turing-Test [131]. Dabei kommuniziert ein Proband mit zwei ihm unbekannten Gesprächspartnern - einer ist ein Mensch und der andere ein Computer. Kann der Proband anhand der Antworten nicht zwischen Mensch und Computer unterscheiden, hat der KI-Algorithmus auf dem Computer den Turing-Test bestanden und kann als intelligent bezeichnet werden. [113] [102]

‚Künstliche Intelligenz' stellt einen Überbegriff für verschiedene Teilbereiche dar. Machine Learning (ML) ist ein zentrales Anwendungsfeld der KI und seinerseits eine Überkategorie für eine Vielzahl von Ansätzen, die darauf abzielen, Muster und Regelmäßigkeiten in vorhandenen Datensätzen zu erkennen und Lösungen für gestellte Probleme zu entwickeln (vgl. Abbildung 2.8). Das Hauptziel von Machine Learning ist, Computersysteme zu befähigen, autonom zu lernen. Somit können mit ML-Methoden Lösungswege gefunden werden, die für den Menschen im Vorfeld nicht ersichtlich sind. [83][19]

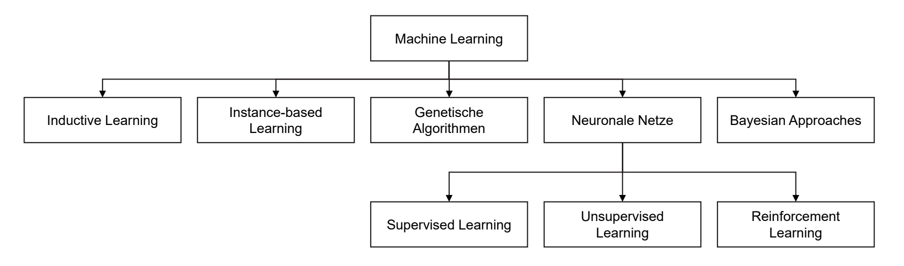


Abbildung 2.8: Hauptgruppen des Machine Learnings [105]

2.4.2 Aufbau eines neuronalen Netzes

Neuronale Netze (NN) bilden eine Untergruppe des Machine Learnings und beschreiben ein spezielles Berechnungsmodell, das dem menschlichen Nervensystem ähnelt. Mehrere Ebenen von Neuronen sind untereinander vernetzt und verarbeiten so eine Eingabe (en: Input) zu einer Ausgabe (en: Output). Jede Ebene

des NN besteht aus prinzipiell gleich aufgebauten Neuronen, die mit den Neuronen der nachfolgenden Ebene verbunden sind. Jedes Neuron erhält, abhängig von seinen Verbindungen, einen oder mehrere Input-Werte, verarbeitet diese intern und gibt sie als Output an die nachfolgenden Neuronen zur weiteren Verarbeitung weiter. [46] [19] [60]

In Abbildung 2.9 wird die Funktionsweise eines Neurons dargestellt.

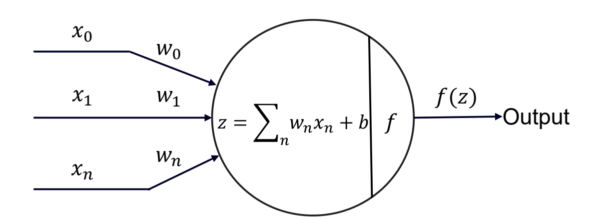


Abbildung 2.9: Aufbau und Funktionsweise eines Neurons i. A. a. [4]

Die Eingangsvariablen $x_1, x_2, ..., x_n$ werden jeweils mit einer spezifischen Gewichtung (en: weight) $w_1, w_2, ..., w_n$ multipliziert. Anschließend werden die Produkte summiert und eine Verzerrung (en: bias) addiert - der errechnete Aktivierungswert z wird einer Aktivierungsfunktion $f(z)$ übergeben. Mit Hilfe der Aktivierungsfunktion $f(z)$ kann ein Output-Wert berechnet und ausgegeben werden. [144]

Die spezifischen Gewichtungen und die Verzerrung sind die Parameter, die beim Lernen des neuronalen Netzes angepasst werden. Die Aktivierungsfunktion ist eine mathematische Funktion, bei der der Zusammenhang von Input- und Output nichtlinear ist. Abhängig davon, wie groß der Aktivierungswert ist, wird ein Output unterschiedlicher Stärke ausgegeben. Typische Aktivierungsfunktionen sind Sigmoid, Tangens-Hyperbolicus, Softmax und ReLU (Rectifier Linear Unit) [19]. Die Aktivierungsfunktionen der Neuronen einer Netzebene sind dabei immer gleich - wobei sich die Aktivierungsfunktionen der einzelnen Netzebenen unterscheiden können. Die verschiedenen Aktivierungsfunktionen sind in der Literatur ausführlich beschrieben [144]. [97] [19]

Ein neuronales Netz besteht grundlegend aus drei Arten von Ebenen - der Input-Ebene, den Netz-Ebenen und der Output-Ebene (vgl. Abbildung 2.10). Informationen werden dem NN über die **Input-Ebene** übergeben. Die Informationen können, abhängig vom vorliegenden Problem, in Form verschiedener Rohdaten oder bereits vor-bearbeiteter Daten vorliegen. Die Anzahl der Input-Neuronen muss dabei mit jener der Eingangsvariablen übereinstimmen. [4]

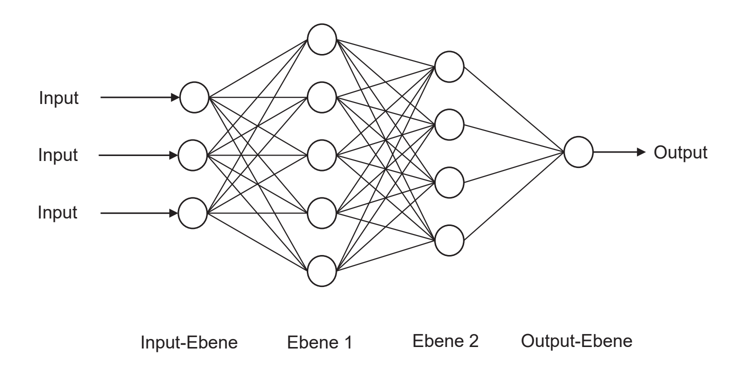


Abbildung 2.10: Architektur eines neuronalen Netzes

Die Anzahl der **Netz-Ebenen** (en: hidden layers) hängt vom zu lösenden Problem ab und basiert in der Regel auf Erfahrungswerten. Mathematisch ist bewiesen, dass grundsätzlich eine Netz-Ebene reicht, um eine beliebige Zielfunktion genau zu approximieren [60]. In der Praxis zeigt sich aber, dass neuronale Netze mit mehreren Netz-Ebenen komplexe Probleme, z. B. bei der Mustererkennung, besser lösen können [15]. Unabhängig davon, aus wie vielen Netz-Ebenen das NN besteht, dient der Output der einzelnen Neuronen einer Netz-Ebene so lange als Input für die nachfolgenden Netz-Ebenen, bis die Output-Ebene erreicht ist. Die **Output-Ebene** kann aus einem oder mehreren Neuronen bestehen. Der Aktivierungswert der Neuronen dieser Ebene ist der Output des neuronalen Netzes. [46]

Vor dem eigentlichen Lern-Prozess werden die spezifischen Gewichtungen und Verzerrungen aller Neuronen zufällig initialisiert. Während des Lernvorgangs werden die beiden Parameter mittels der sogenannten ‚Backpropagation' so angepasst, dass das ‚Loss' (die Differenz zwischen erwartetem und realem Output) möglichst klein wird. Für die Backpropagation gibt es verschiedene Methoden, die ‚Optimizer' genannt werden [144]. [6]

2.4.3 Supervised Learning

Die am weitesten verbreitete Machine-Learning-Methode ist das Supervised Learning (SL). Angelernt wird diese Methode mit gelabelten Trainingsdatensätzen. Mit Hilfe der gelabelten Trainingsdatensätze approximiert diese Methode eine Funktion f^d, die der Zielfunktion f sehr nahe kommt oder diese sogar komplett abbildet [144]. Die Funktion f^d kann nach dem Anlernen auf unbekannte und ungelabelte Datensätze angewendet werden. Das Anlernen der Supervised-Learning-Methode entspricht der Induktion, d. h. dem Schluss vom Einzelfall auf eine allgemeine Erkenntnis. Die Anwendung der Methode entspricht der Deduktion, d.h. dem Rückschluss vom Allgemeinen auf den Einzelfall. Für das Supervised Learning können neuronale Netze eingesetzt werden. Zusätzlich gehören auch Entscheidungsbäume, wie der Naive Bayes und der K Nearest Neighbor-Algorithmus zu dieser Methode. [65] [19]

Für das Anlernen werden Daten benötigt, die zum einen gelabelt werden und zum anderen möglichst viele Fälle abdecken bzw. sich möglichst stark voneinander unterscheiden müssen. Ziel ist es dabei Datenauszüge aus dem gesamten Lösungsraum für das Anlernen zu verwenden, um eine unvoreingenommene (en: unbiased) Funktion f^d zu ermitteln. [113]

Im Unterschied zum Supervised Learning werden beim Unsupervised Learning keine gelabelten Daten benötigt. Der häufigste Anwendungsfall dieser Methode ist die Clusteranalyse, bei der Datensätze auf Basis von Merkmalen gruppiert werden (vgl. Abschnitt 2.6). [19]

2.4.4 Reinforcement Learning

Reinforcement Learning (RL) ist eine ML-Methode, bei der ein Lernen aufgrund einer Interaktion eines Agentens mit seiner Umgebung (en: Environment) erfolgt. Ein Agent wird wie folgt definiert:

„*[...] an agent is a computer system, situated in some environment, that is capable of flexible autonomous action in order to meet its design objectives. Situatedness, in this context, means that the agent receives sensory input from its environment and that it can perform actions which change the environment in some way.*“ [64]

Ähnlich wie beim Lernverhalten von Lebewesen interagiert ein Agent mit einem Environment und zieht aufgrund der Reaktion dessen Rückschlüsse auf sein eigenes Handeln. Über mehrere Iterationen kann der Agent sein Verhalten bzw. seine Strategie (en: Policy) anpassen und damit verbessern. Für die RL-Methode sind vier Grundbausteine notwendig: der Agent, die Action (de: Handlung), das Environment und der Reward (de: Belohnung) [127]. In welchem Verhältnis diese vier Grundbausteine zueinander stehen und welchem zeitlichen Verlauf die Interaktion folgt, wird durch den Markov Decision Process (MDP) festgelegt (vgl. Abbildung 2.11). Der MDP sagt aus, dass die Wahrscheinlichkeit, den Zustand (en: State) s_{t+1} zu erreichen, nur von Zustand s_t abhängt und nicht von weiteren Vorgängern von s_t. Alle RL-Anwendungen müssen die Prämissen des MDP-Entscheidungsproblems erfüllen, um lösbar zu sein. [127] [83]

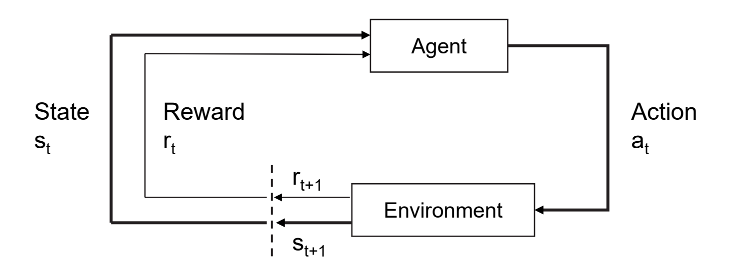


Abbildung 2.11: Agenten-Environment-Interaktion in einem MDP [127]

Das Environment definiert den Rahmen, in dem sich der Agent bewegt. Es kann sich dabei um eine Simulation, ein Videospiel oder auch die reale Welt handeln. Nachdem das Environment eine Action a_t des Agenten erhalten hat, generiert es einen neuen State s_{t+1} und schickt diesen dem Agenten als Antwort auf seine Action. Der Agent ist eine intelligente Entität, der Actions aufgrund seiner Erfahrungen und des aktuellen States auswählt. Alle möglichen Actions sind Bestandteil des Action-Spaces (de: Handlungsspielraum). Ziel des Agenten ist es, seinen Reward r_t zu maximieren. Dafür ermittelt das Environment nach jeder Action mittels einer Reward-Funktion, wie gut bzw. zielführend die Action war, und gibt den Reward zurück an den Agenten, der daraufhin sein Verhalten anpasst, um den Reward zu maximieren. Innerhalb eines Zeitschrittes t wählt der Agent seine Action a_t auf Basis einer Policy π aus. Die Policy ist somit eine Funktion des States: $\pi(s_t) = a_t$. [127] [83] [6]

Nach mehreren Trainingsepisoden hat der Agent eine Policy gelernt, mit der er in dem trainierten Environment seinen Reward maximieren kann. Diese Policy kann der Agent auch auf noch unbekannte Zustände des Environments übertragen. In diesem Kontext wird von ‚Generalisierung‘ gesprochen [6]. Die Fähigkeit eines Agenten, zu generalisieren, wird in drei Bereiche eingeteilt:

- **Keine Generalisierung:** In diesem Fall wird ein Agent in einem Environment mit einer endlichen Zahl an möglichen Zuständen trainiert. Der Agent durchläuft beim Training alle möglichen Zustände des Environments mehrmals und lernt damit dieses Environment auswendig. Die Lernfähigkeit des Agenten wird im Anschluss nur anhand seines Rewards in den durchlaufenen Zuständen bewertet. [88]
- **Schwache Generalisierung:** Ein Agent lernt in dieser Variante nicht nur eine Policy für ein Environment, sondern für mehrere Environments. Die verschiedenen Environments mit ihren unterschiedlichen Zuständen werden beim Training durchlaufen. Die Lernfähigkeit wird hierbei an einem Set aus zufälligen Zuständen, die der Agent beim Training durchlaufen hat, getestet. Der Agent kann in diesem Fall seine Policy in verschiedenen Environments einsetzen. [88]

- **Starke Generalisierung:** Hierbei wird der Agent auf ein Teilset aller möglichen Zustände eines oder mehrerer Environments angelernt. Er durchläuft beim Training somit nicht alle möglichen Zustände. Für die Ermittlung der Lernqualität werden die Zustände herangezogen, die der Agent beim Training nicht gesehen hat. Eine Generalisierung der gelernten Policy auf unbekannte Zustände ist daher notwendig. Besonders bei sehr großen Environments, bei denen es sehr viele mögliche Zustände gibt, ist diese Fähigkeit erforderlich. Wie im Abschnitt 2.3.1 beschrieben, kann das Terminierungsproblem einer Fertigung äußerst komplex werden. Hinzu kommen exponentiell viele States, wenn z. B. Störungen beim Training mit betrachtet werden können. [88]

Eine spezielle Form des Reinforcement Learnings stellt das Multi-Agent Reinforcement Learning dar [61]. Dabei interagieren mehrere Agenten mit demselben Environment. Die verschiedenen Agenten können in diesem Kontext kooperativ zusammenarbeiten, um Aufgaben gemeinsam zu bewältigen, oder in Konkurrenz zueinander stehen [139]. Im Rahmen der in der vorliegenden Arbeit entwickelten Methode wird ein Multi-Agenten-Ansatz genutzt, bei dem sich die Agenten zwar im selben Environment befinden, aber keine direkte Interaktion zwischen ihnen stattfindet. Aus diesem Grund wird an dieser Stelle nicht näher auf das Multi-Agenten-System eingegangen.

2.4.5 Deep-Q-Network (DQN)

DQN-Algorithmen zählen zu den ersten RL-Algorithmen, die eine menschenähnliche bzw. übermenschliche Performance in ausgewählten Aufgabenbereichen erreichen können [92][93]. Grundlage der DQN-Algorithmen ist das Q-Learning, bei dem für alle möglichen Actions eine sogenannte ‚Look-up-Tabelle‘ erstellt wird. Diese Tabelle wächst mit einer zunehmenden Anzahl von States und Actions an und ist aufgrund ihrer Größe ab einem gewissen Punkt nicht mehr abbildbar. DQN-Algorithmen kommen ohne diese Look-up-Tabelle aus und sind deshalb auch für komplexe Probleme anwendbar. [136]

Im Gegensatz zu ‚Policy-based'-Algorithmen, bei denen das NN die beste Action ausgibt, ist das DQN ‚value-based' und gibt somit den zu erwartenden zukünftigen Reward bzw. die Qualität der Actions zurück. Die Qualität Q kann allgemein als Funktion eines gegebenen States $s \in S$ und der gewählten Action $a \in A$ beschrieben werden, siehe Formel 2.5. Der Reward für die Action a im State s wird definiert als $r(s, a)$. Mit dem ‚Discount-Faktor' $\gamma \in [0, 1]$ wird festgelegt, wie stark zukünftige Rewards in die Berechnung von Q miteinbezogen werden sollen. Konvergiert γ gegen null, werden zukünftige Rewards nicht betrachtet; d. h., der Q-Wert wird nur für die nächste Action maximiert. Bei γ-Werten, die gegen eins konvergieren, werden zukünftige Rewards sehr stark gewichtet. In diesem Fall wird nicht der Q-Wert der nächsten Action maximiert, sondern der Q-Wert über einen längeren Zeitraum. [127] [93]

$$Q(s, a) = r(s, a) + \gamma max_a Q(s', a') \tag{2.5}$$

Mittels mehrerer Iterationsschritte kann eine optimale Q-Funktion $Q^*(s, a)$ ermittelt werden. Dabei gilt die Annahme, dass $Q_i(s, a)$ als Q-Funktion in der i-ten Iteration als optimal anzusehen ist: $Q_i \rightarrow Q^*$, wenn $i \rightarrow \infty$. [56]

Der iterative Ansatz, die Q-Funktion zu bestimmen, findet in der Praxis selten Anwendung, da für jedes State-Action-Paar eine neue, unabhängige Q-Funktion ermittelt werden muss. Alternativ kann die Q-Funktion auch approximiert werden. Dafür werden lineare und nichtlineare Funktionsapproximatoren genutzt. [92]

In der vorliegenden Arbeit werden im Zusammenhang mit dem DQN ausschließlich nichtlineare Funktionsapproximatoren wie NNe eingesetzt. Es gilt: $Q(s, a, \theta) \approx Q^*(s, a)$. Die Netzwerkparameter (vgl. Abschnitt 2.4.2) werden als θ definiert. Beim Training des Q-Networks werden die Parameter θ_i zum Zeitpunkt i so angepasst, dass die Loss-Funktion zum Zeitpunkt i minimiert wird (vgl. Formel 2.6) [56]. Die angepassten Netzparameter der vorherigen Iteration werden darin als θ_i^- bezeichnet. [56]

$$L_i(\theta_i) = \mathbb{E}_{s,a,r}\left[(\mathbb{E}'_s(r + \gamma max_{a'} Q(s', a', \theta_i^-)) - Q(s, a, \theta_i))^2\right] \tag{2.6}$$

Ein Vorteil des DQNs ist die Modell-Unabhängigkeit (en: model-free). Modellbasierte Algorithmen besitzen ein festgelegtes Wissen über das Environment, z. B. grundlegende Regeln und Randbedingungen. Das führt dazu, dass modellbasierte Algorithmen schneller lernen, da der Lösungsraum eingeschränkt wird. Das DQN dagegen ist Modell-unabhängig. Explizites Wissen über das Environment muss nicht vorliegen, was zur Folge hat, dass dieser Algorithmus ohne manuellen Aufwand in gänzlich verschiedenartigen Environments antrainiert werden kann. Gerade in der Fertigung, bei der sich der Fertigungsablauf stark von Hersteller zu Hersteller und von Bauteil zu Bauteil unterscheiden kann, ist diese Eigenschaft notwendig.

Beim Training greift der Agent, das NN, auf einen ‚Experience Buffer' (de: Erfahrungsspeicher) zu (vgl. Abbildung 2.12). Alle durchlaufenen States, die gewählten Actions und der dazugehörige Reward werden darin abgespeichert. Der Experience Buffer besteht aus t Einträgen $e_t = (s_t, a_t, r_t, s_{t+1})$ in einem Datenset $D_t = \{e_1, \cdots, e_t\}$. Aufgrund der Limitierung der Berechnungszeit können beim Training nicht alle Einträge e_t betrachtet werden. Stattdessen werden zufällig Einträge für das Training ausgewählt. Bei der Berechnung des Loss wird neben den zufällig ausgewählten Einträgen aus dem Experience Buffer zusätzlich eine vordefinierte Anzahl der letzten State-Action-Tuples mitbetrachtet. Beim Training wird somit auf Offline-Erfahrungen aus dem Experience Buffer und Online-Erfahrungen des Agenten zurückgriffen. Nach der Ermittlung des Loss $L_i(\theta_i)$ werden die Netzwerk-Parameter θ so angepasst, dass das Loss minimiert wird, und anschließend an den Agenten übergeben, der jetzt mit den angepassten Netzwerk-Parametern neue Erfahrungen sammelt. Diese Trainingsschleife wird so lange durchlaufen, bis der Reward des Agenten gegen einen maximalen Wert konvergiert. [56]

Die Qualität der Experience-Buffer-Einträge wirkt sich maßgeblich auf die Performance des DQNs aus. Im weiteren Verlauf der Arbeit ist der Experience Buffer ein wesentlicher Bestandteil der entwickelten Methode. Neben online selbst gewonnenen Erfahrungen kann dieser auch mit offline gesammelten, also externen Einträgen, befüllt werden, was die Kombination verschiedener Methoden zur Erfahrungsgewinnung zulässt.

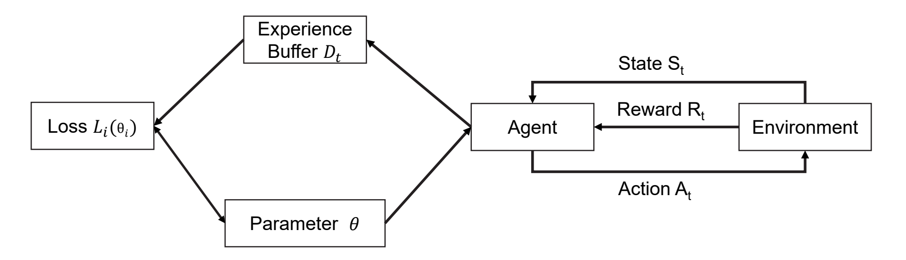


Abbildung 2.12: Schematische Darstellung des Trainings eines DQNs

2.5 Entscheidungsbäume

In diesem Abschnitt werden die Grundlagen von Entscheidungsbäumen erklärt und es wird speziell auf die Monte-Carlo-Tree-Search eingegangen, die einen Teil der nachfolgend entwickelten Methode darstellt. Zusätzlich wird die Kombinationsmöglichkeit der Monte-Carlo-Tree-Search und neuronalen Netzen erläutert.

2.5.1 Aufbau und Funktionsweise eines Entscheidungsbaums

Ein Entscheidungsbaum ist eine Visualisierung von Entscheidungsregeln in einem mehrstufigen Entscheidungsprozess [94]. Startpunkt dafür ist ein Ausgangsknoten, von dem aus mehrere Actions ausgeführt werden können und somit neue Unterknoten bzw. States, auch ‚Kinder' genannt, hervorgehen. Dieser hierarchische Aufbau kann sich, abhängig von der Komplexität des Entscheidungsprozesses, beliebig weit fortsetzen. Da Entscheidungsbäume schon bei verhältnismäßig kleinen State- und Action-Lösungsräumen extrem groß werden können, wurden Such-Algorithmen entwickelt, die den Entscheidungsbaum sequenziell aufbauen und nur dort weitere Knoten anlegen, wo dies sinnvoll ist [94]. Beispiele für diese Art von Such-Algorithmen sind A* und $\alpha\beta$. Um einen Knoten bewerten zu können, brauchen beide Such-Algorithmen eine heuristische Positionsbewertungsfunktion, die in Abhängigkeit vom Anwendungsfall entwickelt werden muss.

Für die Bewertung einzelner Knoten gibt es verschiedene Strategien (vgl. Abbildung 2.13). [20]

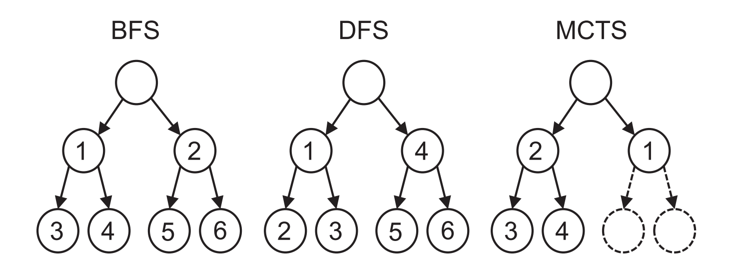


Abbildung 2.13: Reihenfolge der Knoten-Abarbeitung bei verschiedenen Such-Strategien [141]

Bei der Breadth-First-Search (BFS) (de: Breitensuche) werden erst die Knoten derselben Tiefe analysiert - die Suche findet also in der Breite statt. Die Depth-First Search (DFS) (de: Tiefensuche) durchforscht einen Teil des Entscheidungsbaums komplett, bevor ein weiterer Teilbaum untersucht wird - die Suche vollzieht sich hier in der Tiefe. Der Nachteil dieser beiden Strategien ist die vorgegebene Reihenfolge der Knoten-Abarbeitung. Hat ein Entscheidungsbaum eine große Tiefe, wird die Ermittlung des optimalen Pfades mit der BFS-Strategie sehr lange dauern, da alle vorgelagerten Knoten berechnet werden müssen. Handelt es sich um einen Entscheidungsbaum, bei dem ein Teilbaum sehr groß ist und sich besser bewertete Knoten in einem anderen Teilbaum befinden, wird die Berechnung mit der DFS-Strategie eine unnötig lange Rechenzeit mit sich bringen. Die Monte-Carlo-Tree-Search (MCTS) untersucht die Knoten in einer nicht vorbestimmten Reihenfolge [26]. Für jeden Knoten wird ein Monte-Carlo-Wert berechnet, anhand dessen Knoten miteinander verglichen werden können; die MCTS folgt somit immer dem für sie wertvollsten Knoten. Im Gegensatz zu den Such-Algorithmen A* und $\alpha\beta$ benötigt die MCTS keine heuristische Positionsbewertungsfunktion, da bei der MCTS der Lösungsraum zufällig erkundet wird und Knoten mittels der Monte-Carlo-Simulation bewertet werden (vgl. Kapitel 2.5.2). [20] [49]

2.5.2 Monte-Carlo-Tree-Search (MCTS)

Die Monte-Carlo-Tree-Search ist ein heuristischer Suchalgorithmus, dessen Suche auf Grundlage der **Monte-Carlo-Simulation** erfolgt. Bei der Monte-Carlo-Simulation wird ein State in einem Spiel evaluiert, indem von diesem State aus zufällige, endliche Simulationen (en: rollouts) ausgeführt werden. Der State selbst wird eindeutig in Form eines State-Action-Paares beschrieben als $S(s, a)$. Nach jedem Rollout i erfolgt dessen Bewertung in Form einer Payoff-Variable z_i. Den Durchschnitt nach $N(s, a)$ Rollouts bildet die Bewertung des jeweiligen Zustands $Q(s, a)$, der auch Monte-Carlo-Wert genannt, und wie in Formel 2.7 berechnet wird. [14][43]

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s)} I_i(s,a) z_i \tag{2.7}$$

In Formel 2.7 ist $I_i(s, a)$ eine Indikatorfunktion, die 1 ausgibt, wenn in der Simulation i Action a gewählt wird, und 0 bei einer anderen Action [43]. Diese Unterscheidung ist wesentlich, da mehrere Rollouts des States s stattfinden, aber bei nicht allen Action a gewählt wird.

Die **Monte-Carlo-Tree-Search** nutzt die Monte-Carlo-Simulation, um die einzelnen Knoten zu evaluieren. Die Monte-Carlo-Werte für einen Knoten und die unterschiedlichen Actions werden dann verglichen, um die beste Action für diesen Knoten auszuwählen. Für die Auswahl der Knoten durchläuft die MCTS die vier Iterationsschritte Selektion, Expansion, Simulation und Backpropagation, die nachfolgend genauer erklärt werden (vgl. Abbildung 2.14). [26]

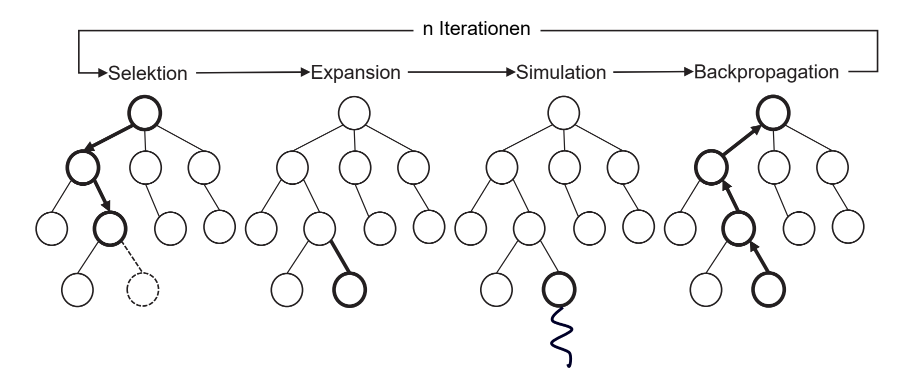


Abbildung 2.14: Iterationsschritte der MCTS [26]

Im ersten Schritt erfolgt die **Selektion**. Dabei werden, ausgehend vom Ausgangsknoten, einzelne Unterknoten (en: children) so lange mittels einer Selektionsstrategie durchlaufen, bis ein End-Knoten (en: leaf) erreicht wird, dessen Unterknoten noch nicht bekannt sind. Die Selektionsstrategie gibt das Verhältnis zwischen der Erkundung neuer Knoten (en: exploration) und der Nutzung bekannter Knoten (en: exploitation) vor. Einerseits ist es sinnvoll, einem bekannten Pfad zu folgen, der Knoten mit vergleichbar hohen Monte-Carlo-Werten besitzt; anderseits können unentdeckte Pfade Knoten mit noch besseren Monte-Carlo-Werten enthalten. In der Literatur wird die Balance zwischen Exploration und Exploitation im Zusammenhang mit dem Multi-Armed-Bandit-Problem erforscht [67]. Die Selektion eines Knotens kann als Multi-Armed-Bandit-Problem angesehen werden, da für die Erweiterung des Pfades mehrere Knoten zur Auswahl stehen, die unterschiedliche und unbekannte Monte-Carlo-Werte aufweisen. Basierend auf vergangenen Durchläufen soll jetzt der beste Knoten ausgewählt werden. Der Hauptunterschied zwischen dem Selektionsproblem der MCTS und dem Multi-Armed-Bandit-Problem ist jedoch die rekursive Auswahl mehrerer Knoten bei der MCTS. [52]

Die Selektion eines Unterknotens k erfolgt nach der Formel der Upper Confidence Bound for Trees (de: obere Konfidenzgrenz) für Entscheidungsbäume), vgl. Formel 2.8. [70]

$$k \in argmax_{i \in I} \left(v_i + C \cdot \sqrt{\frac{\ln n_p}{n_i}} \right) \tag{2.8}$$

In Formel 2.8 ist v_i der Knotenwert des Knotens i. Die Anzahl der Besuche von Knoten i wird als n_i definiert. Analog dazu werden die Besuche des aktuellen Knotens p als n_p bestimmt. Die Variable C muss manuell angepasst werden und beeinflusst das Verhältnis zwischen Exploration und Exploitation. Die Anzahl aller auswählbaren Unterknoten des Knotens p wird mit I beschrieben. Es gibt auch andere Strategien wie OMC [27] oder PPBM [31], und auch von der Upper-Confidence-Bound-Formel gibt es mehrere Variationen [44].

Nach der Auswahl eines Knotens in der Selektionsphase erfolgt im nächsten Schritt die **Expansion** des Knotens. Einer oder mehrere der neuen Unterknoten werden in diesem Schritt der MCTS hinzugefügt. Wie viele Unterknoten angelegt werden, hängt von der Expansions-Strategie ab. Da nicht der gesamte Entscheidungsbaum abgespeichert werden kann, wird in der Regel pro Simulation ein Unterknoten abgespeichert, der zufällig ausgewählt wird. [26]

Bei der **Simulation**, dem dritten Schritt der MCTS-Iteration, wird eine Monte-Carlo-Simulation ausgeführt. Von dem in der Expansionsphase ausgewählten neuen Unterknoten werden mehrere Rollouts durchgeführt. Für die Rollouts gibt es verschiedene Strategien. So kann die Auswahl der folgenden Knoten rein zufällig geschehen, bis ein Endzustand (en: terminal state) erreicht ist. Die Zufallsstrategie ist nicht rechenaufwendig und lässt deshalb viele Rollouts in kurzer Zeit zu. Die zufälligen Entscheidungen dieser Strategie führen aber unter Umständen zu schwachen bzw. unrealistischen Ergebnissen, was die Lösungsqualität der MCTS negativ beeinflusst. Eine Heuristik mit Domänenwissen kann auch für die Rollouts eingesetzt werden und verbessert deren Ergebnisse in der Regel deutlich, benötigt aber auch mehr Rechenzeit was sich negativ auf die Anzahl der Simulationen pro

Sekunde auswirkt. Auch hier können sich die Ergebnisse der MCTS verschlechtern. Eine gute Rollout-Strategie ist immer ein Abwägen zwischen der Anzahl der möglichen Rollouts und der Genauigkeit der Rollout-Ergebnisse. [26]

Der vierte Schritt bei einer MCTS-Iteration ist die **Backpropagation**. Das erzielte Ergebnis aus der Simulationsphase wird dabei über die besuchten Knoten bis zum Ausgangsknoten zurückgegeben, und deren Knoten-Werte v_i werden in der Upper-Confidence-Bound-Formel aktualisiert. Nach mehreren MCTS-Iterationen und mehreren neuen gefundenen Unterknoten wird einer dieser Unterknoten der MCTS dauerhaft zugeordnet. Es gibt verschiedene Möglichkeiten, diesen Unterknoten auszuwählen (vgl. Tabelle 2.2).

Tabelle 2.2: Auswahlkriterien bei der Wahl der Unterknoten, i. A. a. [28]

Auswahlmethode	Definition
Max Child	Das Max Child ist der Unterknoten mit dem höchsten Knotenwert.
Robust Child	Das Robust Child ist der Unterknoten, der am meisten besucht wurde.
Robust-max Child	Das Robust-max Child ist der Unterknoten, der sowohl am meisten besucht wurde als auch den höchsten Knotenwert aufweist.
Secure Child	Das Secure Child ist der Unterknoten, der die untere Konfidenzgrenze maximiert.

Die Vorteile der MCTS können mit den Stichworten **aheuristic** (de: unheuristisch), **anytime** (de: jederzeit) und **asymmetric** (de: unsymmetrisch) zusammengefasst werden [21]. Die MCTS ist unheuristisch, da sie ohne Domänenwissen in jeder Domäne eingesetzt werden kann, die sich mit einem Entscheidungsbaum abbilden lässt. Erfolgreiche Implementierungen wie bei AlphaGo zeigen jedoch, dass die MCTS mit Domänenwissen deutlich verbessert werden kann. Für jede Anwendung ist daher ein Abwägen zwischen schnellen Ergebnissen, die Anwendungen für verschiedene Domänen oder guten Ergebnissen in einer Domäne notwendig. Zusätzlich durchläuft die MCTS die im Vorfeld beschriebenen Iterationen (vgl. Abbildung 2.14) so lange, bis ein Zeit- oder Speicherplatzlimit erreicht wird. Es ist aber möglich, zu jedem Zeitpunkt eine Aktion der MCTS abzufragen. Dieses Ergebnis ist unter Umständen nicht optimal, da längere Rolloutzeiten die Qualität der Ergebnisse verbessern können. Die MCTS ist unsymmetrisch, was bedeutet,

dass nicht alle Knoten gleich oft besucht werden. Knoten bzw. Abschnitte in der MCTS mit einer höheren Wahrscheinlichkeit gute Ergebnisse herbeizuführen werden häufiger besucht, als Knoten mit einer geringen Wahrscheinlichkeit für gute Ergebnisse. Der Entscheidungsbaum wird somit nur teilweise aufgebaut und ist unsymmetrisch - damit unterscheidet sich die MCTS von anderen Entscheidungsbäumen (vgl. Kapitel 2.5.1). [21]

2.5.3 Parallelisierung der MCTS

Die MCTS benötigt während der Simulation bzw. während des Rollouts (vgl. Abschnitt 2.5.2) keine weiteren Informationen des Entscheidungsbaums. Die Rollouts können somit komplett unabhängig voneinander erfolgen. Die Parallelisierung der Rollouts ermöglicht es, mehrere Rollouts gleichzeitig und nicht sequenziell zu durchlaufen. Daher können in kürzerer Zeit mehr Rollouts realisiert werden, was eine Verkürzung der Rechenzeit bei gleich gutem Rollout-Ergebnis bewirkt. Voraussetzung für die Parallelisierung ist ein symmetrisches Multiprozessorsystem, bei dem ein CPU-Thread jeweils einen Rollout durchführt. [79]

Es werden drei Arten der Parallelisierung bei der MCTS unterschieden: die Unterknoten-Parallelisierung, die Ausgangsknoten-Parallelisierung und der Entscheidungsbaum-Parallelisierung (vgl. Abbildung 2.15) [25].

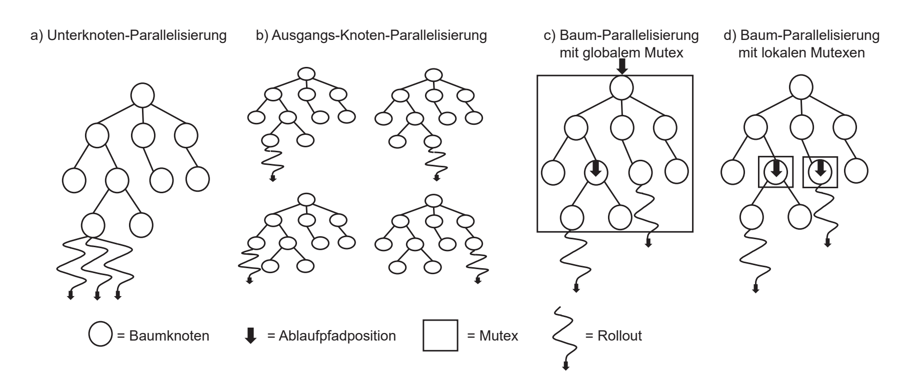


Abbildung 2.15: Möglichkeiten zur Parallelisierung der MCTS [52]

Bei der **Unterknoten-Parallelisierung** (vgl. Abbildung 2.15 a) werden ausgehend von einem Unterknoten simultan mehrere Rollouts durchgeführt. Das Ziel dieser Parallelisierungsmethode ist, einen Unterknoten aufgrund der vielfachen Rollouts besser bewerten zu können. Nach den Rollouts werden die entsprechenden Werte des Entscheidungsbaums entlang des gewählten Pfades während der Backpropagation-Phase angepasst. [25]

Von **Ausgangsknoten-Parallelisierung** (vgl. Abbildung 2.15 b) wird gesprochen, wenn parallel mehrere Entscheidungsbäume aufgebaut werden. Die verschiedenen Entscheidungsbäume teilen sich keine Informationen und sind komplett unabhängig voneinander. Ist die Berechnungszeit abgelaufen, werden die Entscheidungsbäume zusammengeführt und die Werte der einzelnen Knoten addiert. Die beste Aktion hat dann den höchsten Knotenwert. [25]

Anders als bei der Ausgangsknoten-Parallelisierung, bei der der gesamte Entscheidungsbaum parallelisiert wird, werden bei der **Entscheidungsbaum-Parallelisierung** (vgl. Abbildung 2.15 c) nur Baumabschnitte parallelisiert. Dabei wird nur ein Entscheidungsbaum aufgebaut, von dem parallel an mehreren Knoten mehrere Rollouts ausgeführt werden. Da Knoten-Werte korrumpiert werden könnten, weil Baumparameter nach jeder Simulation angepasst werden, ist es

notwendig, Mutexe (en: mutual exclusion) einzuführen. Mutexe grenzen unabhängige Teile des Entscheidungsbaums ab und verhindern so, dass sich die unterschiedlichen Simulationen gegenseitig beeinträchtigen. Bei der Parallelisierung mit einem globalen Mutex wird der gesamte Entscheidungsbaum gesperrt. Hierbei kann nur ein Pfad innerhalb der MCTS abgelaufen werden. Parallel dazu können aber Simulationen anderer Unterknoten erfolgen - damit unterscheidet sich die Entscheidungsbaum-Parallelisierung mit einem globalen Mutex von der Unterknoten-Parallelisierung. Lokale Mutexe ermöglichen das Erstellen mehrerer Ablaufpfade gleichzeitig. [29]

2.5.4 MCTS kombiniert mit Reinforcement Learning

Ist der Lösungsraum, der alle möglichen States umfasst, relativ klein, kann die MCTS mit genügend Rechenzeit einen Entscheidungsbaum aufbauen, der alle möglichen States und deren Monte-Carlo-Werte enthält. Bei Spielen wie Go oder auch bei der Terminierung einer Fertigung mit vielen Maschinen und Bauteilen kann nicht der gesamte Entscheidungsbaum aufgebaut werden, da sich dieser aus zu vielen States bzw. Knoten zusammensetzt. Neuronale Netze können die MCTS beim Lernen und beim Rollout unterstützen. In der Literatur gibt es für die Kombination von MCTS und neuronalen Netzen keine Standard-Architektur. Abbildung 2.16 zeigt zwei Möglichkeiten, die MCTS mit einem NN zu kombinieren.

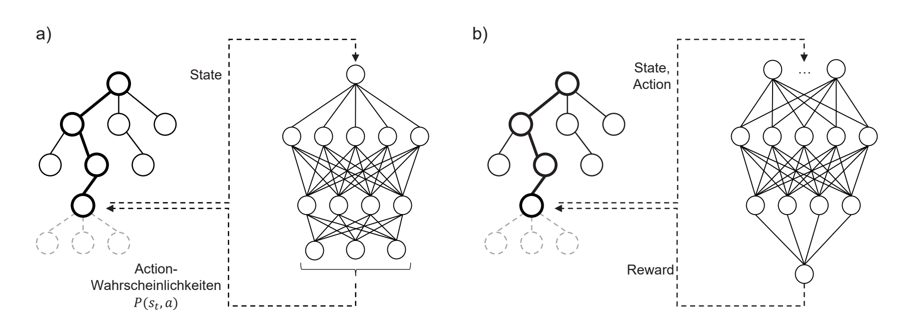


Abbildung 2.16: Kombinationsmöglichkeiten MCTS mit einem NN

In Beispiel (a) der Abbildung 2.16 wird ein NN dazu genutzt, die Action-Wahrscheinlichkeit P zu bestimmen. Als Input für das NN dient in diesem Fall der aktuelle Knoten. Das NN bewertet für den ausgewählten Knoten alle möglichen Actions und deren erwartete Wahrscheinlichkeiten, dass es sich um eine gute Action handelt. In Beispiel (b) der Abbildung 2.16 wird das NN dazu genutzt, den Monte-Carlo-Wert für einen Knoten abzuschätzen. Das NN bekommt als Input den aktuellen Knoten und eine potenzielle Action, die von der MCTS ausgewählt oder wie in Beispiel a von einem NN ermittelt wurde. Ein einziges Output-Neuron gibt dann einen zu erwartenden Monte-Carlo-Wert für diesen Knoten aus. Rollouts sind damit nicht mehr notwendig.

Die Actions während des Rollouts der MCTS können, wie in Abschnitt 2.5.2 beschrieben, zufällig oder mit Hilfe einer Heuristik gewählt werden. Die Entscheidungen im Rollout können aber auch mittels eines trainierten NN getroffen werden. Dabei muss das NN zuvor in dem gegebenen Environment trainiert werden und kann dann nachfolgend im Rollout die vermeintlich beste Aktion auswählen. Im Gegensatz zum zufälligen Rollout kann ein gut trainiertes NN deutlich bessere Entscheidungen über Actions treffen und damit die Qualität der ermittelten Monte-Carlo-Werte verbessern. Der Rollout ist trotzdem schneller, als wenn eine Heuristik verwendet werden würde, denn die Berechnungszeit von neuronalen Netzen ist geringer.

Wenn die MCTS mit einem NN kombiniert wird, werden die Unterknoten nicht mehr zufällig, sondern von einem NN ausgewählt. Das Zusammenwirken MCTS und NN ist in der Formel 2.9 beschrieben, die eine Variante der PUCB-Formel ist [111] [120]. PUCT steht für ‚Polynomial Upper Confidence Trees' auf deutsch ‚obere Polynomialgrenze für Entscheidungsbäume'. Das Acronym UCB bzw. UCT bedeutet ‚Upper-Confidence-Bound' bzw. ‚Upper-Confidence-Bounds applied to Trees' (vgl.Formel 2.8). Während der Selektionsphase (vgl. Abschnitt 2.5.2) wird der entsprechende Knoten nach folgender PUCT-Formel ausgewählt (vgl. Formel 2.9) [120].

$$a_t = \arg\max_a (Q(s_t, a) + U(s_t, a)) \tag{2.9}$$

Der durchschnittliche Reward, der zu erzielen ist, wenn der State (s, a) zum Zeitschritt t mit der Monte-Carlo-Simulation evaluiert wird, wird als $Q(s_t, a)$ bezeichnet (vgl. Formel 2.7). Dabei berechnet sich $U(s_t, a)$ wie in Formel 2.10 [120] dargestellt als:

$$U(s_t, a) = c_{puct} P(s_t, a) \frac{\sqrt{\sum_b N(s_t, b)}}{1 + N(s_t, a)} \tag{2.10}$$

Der Parameter $c_{puct} \in [0; 1]$ beschreibt das Verhältnis zwischen dem Erkunden neuer Knoten und der Nutzung bekannter Pfade bzw. bekannter Knoten. Die Wahrscheinlichkeit, dass von State s aus eine Entscheidung für Action a getroffen wird, ist dabei als $P(s_t, a)$ definiert und wird vom NN ausgegeben (vgl. Abbildung 2.16 (a)). Die Zählvariable N beinhaltet die Anzahl der Besuche des jeweiligen State-Action-Paars. Im Zähler des Bruches steht die Summe aller Zähler aller möglichen Actions b des States s $N(s_t, b)$. Im Nenner des Bruches steht der Zähler $N(s_t, a)$ des State-Action-Paars, das zum Zeitschritt t berechnet wird. [120]

Damit das NN die Action-Wahrscheinlichkeiten bzw. den zu erwartenden Reward einer Action in ausreichender Qualität ausgeben kann, muss das NN trainiert werden (vgl. Abschnitt 2.4.2). Die Trainingsdaten dafür können mit der MCTS selbst erzeugt werden. Der MCTS-Entscheidungsbaum wird in diesem Fall wie in Abschnitt 2.5.2 beschrieben aufgebaut. Für jeden State s_t wird die ausgewählte Action a zum Zeitpunkt t gespeichert. Erreicht die MCTS einen terminalen State, wird der erzielte Reward z (vgl. Abschnitt 2.7) dem State-Action-Paar zugeordnet und zu einem Tripel (s_t, a, z) zusammengefügt. Die Tripel werden als Trainingsdaten für das NN genutzt. Das NN lernt, den zu erwartenden Reward z eines State-Action-Paars (s_t, a) zu approximieren. Dafür wird die Machine-Learning-Methode des Supervised Learnings verwendet (vgl. Abschnitt 2.4.3).

Das State-Action-Paar ist der Input für das NN und der Reward das Label. Auf diese Weise lernt das NN ausgehend von einem State-Action-Paar den Reward zu bestimmen. Das so angelernte NN kann nach dem Training, wie in Abbildung 2.16 (a) dargestellt, die MCTS beim Aufbau eines neuen Entscheidungsbaums unterstützen, sodass die MCTS schneller gute Ergebnisse erreicht. [120] [74][110]

2.6 Clusteranalyse von Datensätzen

Nachfolgend werden die Zielstellung und das Vorgehen bei der Clusteranalyse beschrieben. Die Clusteranalyse ist ein zentraler Bestandteil der in Kapitel 4 vorgestellten Methode. Sie wird dafür genutzt, ein komplexes Problem in besser zu lösende Teilprobleme aufzugliedern.

2.6.1 Ziel der Clusteranalyse

Das Ziel der Clusteranalyse ist, Cluster in einer Datenmenge zu identifizieren. Dabei werden die Daten aufgrund ihrer Ähnlichkeit bzw. der Distanz zueinander unterschiedlichen Clustern zugeordnet. Die Teil-Datenmengen eines einzelnen Clusters weisen daraufhin einen hohen Grad an Ähnlichkeit auf; d. h., die Daten eines Clusters haben eine große Schnittmenge. Die Vorgehensweise bei der Clusteranalyse umfasst folgende drei Schritte [12]:

1. **Bestimmung der Ähnlichkeiten**: Dabei werden zwei Einträge in der Datenmenge miteinander verglichen (vgl. Abschnitt 2.6.2).

2. **Auswahl des Clusteringalgorithmus**: Die verschiedenen Einträge werden aufgrund ihrer Ähnlichkeit unterschiedlichen Clustern zugeordnet (vgl. Abschnitt 2.6.3).

3. **Bestimmung der Clusteranzahl**: Die optimale Clusteranzahl wird hierbei ermittelt (vgl. Abschnitt 2.6.4).

2.6.2 Bestimmung der Ähnlichkeit

Die Bestimmung der Ähnlichkeit von Einträgen in einer Datenmenge ist der erste Schritt der Clusteranalyse. Die Ähnlichkeit bzw. die Distanz von zwei Einträgen kann anhand verschiedener Abstandsfunktionen, sogenannter Metriken, ermittelt werden. Die Distanz zweier Punkte lässt sich über die euklidische Distanz berechnen, mit der der geradlinige Abstand zweier Punkte im n-dimensionalen Raum bestimmt wird. Die Minkowski-Metrik ist eine verallgemeinerte Form der euklidischen Distanz und wird eingesetzt, wenn zwei Einträge quantitativ verglichen werden sollen.

Im Rahmen der vorliegenden Arbeit wird die Jaccard-Metrik genutzt, weil diese im Gegensatz zu den vorher genannten Metriken nicht den Abstand zweier Punkte, sondern die Ähnlichkeit der Einträge ermittelt. Dabei werden zwei Einträge qualitativ auf ihre Ähnlichkeit untersucht. Die Jaccard-Metrik berechnet sich nach Formel 2.11 [5]:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.11}$$

Dabei sind A und B zwei Mengen, die eine Schnittmenge $|A \cap B|$ aufweisen. Alle Einträge der Mengen A und B werden als $|A \cup B|$ bezeichnet. Nachdem die Jaccard-Metrik für alle Einträge in A und B berechnet und daraus eine entsprechende Ähnlichkeitsmatrix erstellt wurde, findet im nächsten Schritt das Clustering statt.

2.6.3 Auswahl des Clusteringalgorithmus

Grundlage für das Clustering ist die Ähnlichkeitsmatrix (vgl. Abschnitt 2.6.2). Zur Auswahl des geeigneten Clusteringalgorithmus stehen zahlreiche unterschiedliche Algorithmen zur Verfügung, die in ihrer Vorgehensweise voneinander abweichen. Umfangreiche Beschreibungen von Clusteringalgorithmen wurden bereits publiziert, z. B. in [138], sodass der Fokus in diesem Abschnitt auf den Clusteringalgorithmus gerichtet wird, der für die nachfolgend entwickelte Methode relevant ist.

Eine Untergruppe der Clusterverfahren bilden die hierarchischen Clusteringalgorithmen. Diese lassen sich weiter in agglomerative und divisive Verfahren unterteilen. Bei den agglomerativen Verfahren wird von der kleinsten Partition, also dem Einzelobjekt, ausgegangen. In den nachfolgenden Schritten des Verfahrens werden diese vielen und sehr kleinen Partitionen in einem größeren Clustern vereinigt. Dazu werden die Distanzen der Cluster zueinander berechnet. Die zwei Cluster mit der größten Ähnlichkeit werden zu einem Cluster zusammengeführt. Dieses Vorgehen wird so lange wiederholt, bis alle Einzelobjekte einem Cluster angehören. Für die Distanzberechnung der Cluster gibt es verschiedene Verfahren auf die an dieser Stelle nicht weiter eingegangen wird [138]. [5] [12]

Im Gegensatz zu den agglomerativen Verfahren bildet beim divisiven Verfahren die gröbste Partition den Ausgangspunkt. Anschließend wird diese Partition in kleine Partitionen aufgeteilt. Beide Verfahren haben den Vorteil, dass die Anzahl der Cluster nicht a-priori definiert werden muss. Zusätzlich muss bei diesen Verfahren, anders als z. B. beim K-Means Clustering [98], kein Startpunkt ausgewählt werden, der einen großen Einfluss auf die Clustererstellung hat.

2.6.4 Bestimmung der Clusteranzahl

Hierarchische, agglomerative Clusterverfahren führen einzelne Partitionen über mehrere Iterationen so lange zusammen, bis eine Zusammenfassung, also eine Partition, aller Einzel-Objekte erreicht ist. Aus diesem Grund muss im nächsten Schritt die Anzahl der sinnvollen Cluster bestimmt werden. Bei der Bestimmung der Anzahl der Cluster muss immer zwischen der Handhabbarkeit (also möglichst wenigen Clustern) und der Anforderung an die Homogenität innerhalb eines Clusters (also möglichst vielen Clustern) abgewogen werden. Ein Mensch ist in der Regel nicht dazu in der Lage, die optimale Clusteranzahl aufgrund logisch begründbarer Vorstellungen zu ermitteln [12].

Methoden, um die optimale Clusteranzahl zu ermitteln, lassen sich in drei Gruppen einordnen: interne, externe und relative Clustervalidierungen. Die interne Clustervalidierung erfasst dabei die Güte der Cluster und die Clusteranzahl mit Hilfe der Informationen der Clusteranalyse. Im Gegensatz dazu werden bei der externen Clustervalidierung externe Informationen (z. B. Label) genutzt um, die Cluster-Güte zu bestimmen. Die Anzahl der Cluster ist dafür in der Regel schon bekannt. Bei der relativen Clustervalidierung werden verschiedene Parameter, z. B. die Anzahl der Cluster, so lange verändert, bis ein Optimum gefunden wird. Nachfolgend wird näher auf Methoden der internen Clustervalidierung eingegangen, da diese Bestandteil der in Kapitel 4 entwickelten Methode sind. [138]

Methoden der internen Clustervalidierung ermitteln in der Regel Indizes, die das Verhältnis zwischen ‚Kompaktheit' und ‚Trennung' angeben. Die Kompaktheit misst dabei wie ähnlich die Objekte in einem Cluster sind. Je höher die Kompaktheit, desto besser ist auch das Clustering. Die Trennung beschreibt, wie weit entfernt bzw. wie unähnlich die Cluster zueinander sind. Ein weit verbreiteter Index der internen Clustervalidierung ist der Silhouetten Index S_{index}, der das Verhältnis zwischen Kompaktheit und Trennung wiedergibt [112] [85]. Der Silhouetten Index S_{index} berechnet sich wie in Formel 2.12 angegeben [112].

$$S_{index}(k) = \frac{1}{m} \sum_{i=1}^{m} \frac{b(i) - a(i)}{\max\{a(i); b(i)\}}, \quad S_{index}(k) \in [-1, 1] \tag{2.12}$$

Dabei repräsentiert k die Anzahl der Cluster und m die Anzahl aller Objekte. Die Funktionen $a(i)$ und $b(i)$ stellen die durchschnittliche Distanz des Objekts i zum nächstgelegenen Cluster A und dem zweit-nächstgelegenen Cluster B dar. Bei jeder Iteration wird i ein anderes Objekt von m zugewiesen, sodass am Schluss alle Distanzen aller Objekte zu den Clustern ermittelt sind. Auf diese Weise wird der Silhouetten-Index für jedes Objekt i berechnet und letztlich der Durchschnitt über alle Silhouetten-Indizes bestimmt. Dieses Vorgehen kann anschließend für alle möglichen Cluster-Anzahlen k wiederholt werden. Von den Silhouetten-Indizes $S_{index}(k)$ aller möglichen k, kann dann dasjenige k ausgewählt werden, das den höchsten Silhouetten-Index aufweist - die optimale Cluster-Anzahl ist damit ermittelt.

3 Stand der Technik in Industrie und Forschung

Dieses Kapitel liefert einen Überblick über den Stand der Technik in Industrie und Forschung im Bereich der Terminierung und insbesondere der dynamischen Terminierung. Als Erstes wird auf die Fertigungsplanung und -steuerung sowie die dafür notwendigen IT-Systeme eingegangen. Daraufhin werden verschiedene Forschungsansätze für die Terminierung vorgestellt, verglichen und auf ihre Anwendbarkeit für die dynamische Terminierung untersucht. Ausgehend von den Anforderungen der Industrie und den Forschungsansätzen für eine flexible Fertigung werden anschließend Anforderungen an die dynamische Terminierung abgeleitet. Am Ende des Kapitels werden bestehende Terminierungsansätze anhand dieser Anforderungen an die dynamische Terminierung eingeordnet und die zu schließende Forschungslücke wird aufgezeigt.

3.1 Fertigungsplanung und Steuerung in der Industrie

An dem Prozess der Fertigungsplanung und -steuerung sind verschiedene IT-Systeme beteiligt. Nachfolgend wird daher zunächst auf die standardisierten Funktions-Hierarchien der Fertigung eingegangen. Im Anschluss daran werden bestehende IT-Systeme diesen Funktions-Hierarchien zugeordnet. Letztere werden nach dem IEC Standard „IEC 62264-1:2013 Enterprise-control system integration“ definiert (vgl. Tabelle 3.1) [63].

Tabelle 3.1: Funktionale Hierarchien in der Fertigung, i. A. a. [63]

Hierarchieebene	Funktionen	Zeithorizont
Level 4: Geschäftsplanung und Logistik	Lang- und mittelfristige Fertigungsplanung, Inventarüberwachung, Materialplanung	Monate, Wochen, Tage
Level 3: Fertigungsmanagement	Kurzfristige Fertigungsplanung, Qualitätsmanagement	Tage, Stunden, Minuten, Sekunden
Level 2: Steuerungsebene	Fertigungsüberwachung	Stunden, Minuten, Sekunden, Millisekunden
Level 1: Sensorebene	Messwerterfassung	-
Level 0: Fertigungsprozess	-	-

- **Level 4** aus Tabelle 3.1 ist für die Abwicklung betriebswirtschaftlicher Prozesse wie der eingehenden Bestellungen, der langfristigen Materialplanung und der lang- und mittelfristigen Fertigungsplanung zuständig. Zusätzlich werden Logistikprozesse innerhalb dieser Funktionsebene gesteuert. Die betrachteten Prozesse dieser Ebene haben einen Zeithorizont von Monaten bis Tagen. [63]

- **Level 3** dient als Datendrehscheibe zwischen den betriebswirtschaftlichen Prozessen von Level 4 und den Fertigungsprozessen auf Level 2 bis 0. Lang- und mittelfristige Fertigungsplanung und Materialbestände werden aus Level 4 bezogen, bearbeitet und an die Fertigung übermittelt. Rückmeldungen aus der Fertigung, z. B. Auftragsrückmeldungen, werden in Level 3 aufgenommen und an IT-Systeme in Level 4 weitergeleitet. Abweichungen von Plan- und Ist-Werten der Fertigung können so im Level 3 erfasst werden. Eine weitere Aufgabe ist die Auftragsterminierung (vgl. Abschnitt 2.2.4). Auf Basis der Planaufträge, der Materialbestände und der Verfügbarkeiten in der Fertigung wird eine detaillierte Terminierung erstellt. Der Planungshorizont liegt zwischen Tagen und Sekunden. [63]

- Die Steuerungsebene im **Level 2** umfasst die Fertigungsüberwachung auf Maschinenebene. Störungen werden hier erfasst und an Level 3 und 4 weitergeleitet. [63]

- **Level 1 und 0** umfassen die Sensoren der Bearbeitungsmaschinen und den Fertigungsprozess selbst. [63]

Die Funktionen der verschiedenen Hierarchieebenen sind für die Fertigung notwendig und können mit unterschiedlichen IT-Systemen realisiert werden. Dabei gibt es Ansätze, bei denen ein IT-System möglichst alle Funktions-Ebenen umfasst. Dieser Ansatz ist besonders in kleineren, mittelständischen Unternehmen verbreitet. Der Vorteil besteht in erster Linie darin, dass Schnittstellen wegfallen, eine einheitliche Benutzeroberfläche gegeben ist und ein geringerer Aufwand bei der IT-System-Wartung erforderlich wird. Diese IT-Systeme besitzen in der Regel einen Funktionsumfang, der speziell auf den Mittelstand abgestimmt ist. [1]

Unternehmen mit der Notwendigkeit einer größeren Anpassungsmöglichkeit und individuellen Anforderungen greifen auf IT-Systeme zurück, die auf einzelne Funktionen wie die Fertigungsterminierung spezialisiert sind. Anstatt eines einzelnen IT-Systems ergibt sich hier eine IT-Systemlandschaft mit verschiedenen IT-Systemen und deren Schnittstellen. Der Implementierungsaufwand und die Wartungsintensität einer solchen IT-Systemlandschaft sind entsprechend höher. Da sich der Funktionsumfang der IT-Systeme stark unterscheidet, gibt es keine einheitliche IT-Systemlandschaft für die Fertigung. Grundsätzlich lassen sich aber, mit Bezug auf Fertigungsplanung und -steuerung, drei Arten von IT-Systemen unterscheiden [90]:

- Enterprise-Resource-Planning (ERP)
- Manufacturing-Execution-System (MES)
- Advanced-Planning-System (APS)

Die dazugehörige IT-Systemlandschaft ist in Abbildung 3.1 dargestellt.

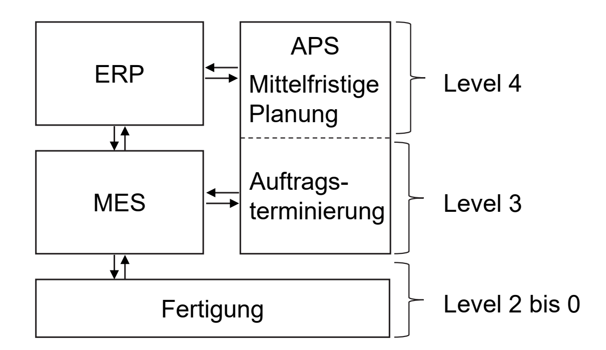


Abbildung 3.1: Einordnung von ERP, MES und APS in die funktionalen Hierarchien der Fertigung, i. A. a. [90]

Das ERP-System ist ein betriebswirtschaftliches IT-System, das eine Vielzahl von operativen und dispositiven Geschäftsprozessen eines Unternehmens unterstützt. Material- und Ressourcenplanung gehören ebenfalls zum typischen Funktionsumfang von ERP-Systemen. Auch wenn ihr Funktionsumfang variiert, erfüllen IT-Systeme dieser Art hauptsächlich Funktionen der Hierarchieebene 4. Die Funktion der mittelfristigen Planung besitzen sowohl ERP- als auch APS-Systeme [73]. Die kurzfristige Planung bzw. die Terminierung der FAUFs (vgl. Abschnitt 2.2.4) ist Aufgabe des Fertigungsmanagements auf Level 3 und erfolgt meist in dafür konzipierten IT-Systemen. In Tabelle 3.2 wird ein Vergleich gängiger APS-Systeme und deren Funktionalitäten aufgelistet. Alle dabei untersuchten APS-Systeme bieten Funktionen für die lang-, mittel- und kurzfristige Planung. Bei einer Abweichung von der Fertigungsplanung, z. B. aufgrund von Störungen, sieht keines der aufgelisteten APS-Systeme eine echtzeitnahe Reaktion vor. Tritt eine Störung auf, greift daher ein Mitarbeiter als Steuerer in den Fertigungsablauf ein und priorisiert FAUFs um und/oder ändert die geplanten Maschinen/Arbeitsplätze der FAUFs (vgl. Abschnitt 2.2.5). Die Eingriffe basieren dabei auf dem Erfahrungswissen des Steuerers und persönlichen Absprachen in der Fertigung [51]. Aktuell gibt es kein IT-System, das die Komplexität der Terminierungsproblematik bewältigen kann und gleichzeitig eine schnelle Reaktion im Störungsfall sicherstellt. Eine Neuterminierung ist immer zeitaufwendig und muss manuell angestoßen werden.

Tabelle 3.2: Vergleich von APS-Systemen

APS-System	Modulname	Modulfunktion
Aspen Plant Scheduler [10]	Plant Scheduling	Lang- und mittelfristige Planung
	Plant Scheduler - Extended Optimization	Kurzfristige Planung
SAP Advanced Planner and Optimizer (APO) [11]	Production Planning and Detailed Scheduling (PP/DS)	Lang-,mittel- und kurzfristige Planung
Oracle JD Edwards EnterpriseOne [99]	Master Production Scheduling (MPS)	Lang-, mittel- und kurzfritige Planung
Siemens Opcenter [119]	APS	Lang-, mittel- und kurzfristige Planung
ABAS ERP [1]	ABAS APS	Lang-, mittel- und kurzfristige Planung

Neben der lang- und mittelfristigen Planung und der Terminierung muss eine neue Terminierungs-Ebene definiert werden, die die Terminierung der FAUFs im Störungsfall automatisch anpasst. Die neue Terminierungs-Ebene wird im Rahmen dieser Arbeit als **‚dynamische Terminierung‘** definiert. Die Berechnungszeit der dynamischen Terminierung darf, im Gegensatz zur klassischen Terminierung, nur wenige Minuten betragen. Der Planungshorizont ist abhängig von den Prozesszeiten, muss aber die Zeit bis zur nächsten Neuterminierung überbrücken. In einem Zwei-Schicht-Betrieb muss die dynamische Terminierung im Falle einer Störung zu Beginn der ersten Schicht maximal zwei Schichten berücksichtigen, da eine Neuterminierung nach dem Ende der zweiten Schicht angestoßen werden kann. Da der Planungshorizont der dynamischen Terminierung viel kleiner als jeder der klassischen Terminierung ist, nimmt die Komplexität des Terminierungsproblems ab, was eine schnelle Ad-hoc-Lösung ermöglicht. Dabei ist die dynamische Terminierung bei den funktionalen Hierarchien in Tabelle 3.1 auf dem Level 3 ‚Fertigungsmanagement‘ angesiedelt.

3.2 Stand der Forschung zu Lösungsmöglichkeiten für Terminierungsprobleme

Die in Abschnitt 2.3.1 vorgestellten Terminierungsprobleme können auf unterschiedliche Art und Weise gelöst werden. Nachfolgend wird auf die gängigen Methoden eingegangen. Zusätzlich werden relevante Veröffentlichungen zu den jeweiligen Lösungsmethode vorgestellt und deren Anwendbarkeit für die dynamische Terminierung untersucht.

3.2.1 Exakte Lösungsmethoden

Exakte Lösungsmethoden finden immer die exakte Lösung für ein Optimierungsproblem. Die Zeit zum Lösen des Optimierungsproblems hängt von dessen Größe bzw. Komplexität ab. Die in Abschnitt 2.3.1 vorgestellten Optimierungsprobleme JSSP und OSSP sind beide NP-schwer. Besonders bei Optimierungsproblemen mit einem großen Lösungsraum führt das zu Rechenzeiten, die für dynamisch auftretende Aufgaben unangemessen lang sind. Die Familie der exakten Lösungsmethoden für Terminierungsprobleme umfasst unter anderem Branch-and-Bound-Algorithmen [9] [22] und Mixed-Integer-Programming [71]. [140]

3.2.2 (Meta-)Heuristiken

Heuristiken sind problemspezifische Lösungsmethoden, die Optimierungsprobleme nicht exakt lösen. Sie sind regelbasiert und dazu in der Lage, in Echtzeit Entscheidungen aufgrund von Produktprioritäten o. Ä. zu treffen. Die Komplexität der Regeln hängt stark vom zu optimierenden System ab. Ein Beispiel für eine Heuristik ist die Shortest-Processing-Time (SPT) Heuristik [86]. Dabei wird das Bauteil in der Warteschlange einer Maschine als nächstes bearbeitet, das die kürzeste Bearbeitungszeit aller wartenden Bauteile besitzt. Bei komplexeren

Zusammenhängen werden demzufolge auch die Heuristiken komplexer. Für die Definition der Heuristiken ist entsprechendes Domainwissen notwendig, was die Erstellung und Anpassung dieser Regeln erschwert. [36]

Metaheuristiken sind im Gegensatz zu klassischen Heuristiken nicht problemspezifisch. Sie unterstützen lokale Such-Heuristiken, z. B. die ‚Variable Neighborhood Search', um nicht in einem lokalen Optimum gefangen zu bleiben [91]. Bei der Variable Neighborhood Search startet die Suche nach einer besseren Lösung bei einer gegebenen Lösung. Iterativ wird im nächsten Schritt nach einer besseren Lösung in der Nachbarschaft gesucht. Wird keine bessere Lösung gefunden, bricht die Suche ab - das kann der Fall sein, wenn ein lokales Optimum gefunden wurde. Metaheuristiken wie die ‚Tabu Search' oder das ‚Simulated Annealing' unterstützen die Such-Heuristik entweder bei der Auswahl der Ausgangslösung oder bei der Suche selbst [100]. Die Such-Heuristik findet so bessere Lösungen. In vielen wissenschaftlichen Veröffentlichungen zeigen Metaheuristiken, dass sie das statische Terminierungsproblem gut lösen können [124]. Umfängliche Untersuchungen zur Frage, ob diese Methoden auch für die dynamische Terminierung genutzt werden könnten, sind bisher nicht zu recherchieren.

3.2.3 Genetische Algorithmen

Als genetische Algorithmen (GAs) werden Lösungsverfahren bezeichnet, die sich am Vorbild der biologischen Evolution orientieren [47]. Sie bilden dabei eine Sonderform der Metaheuristiken. GAs lösen Optimierungsprobleme iterativ (vgl. Abbildung 3.2).

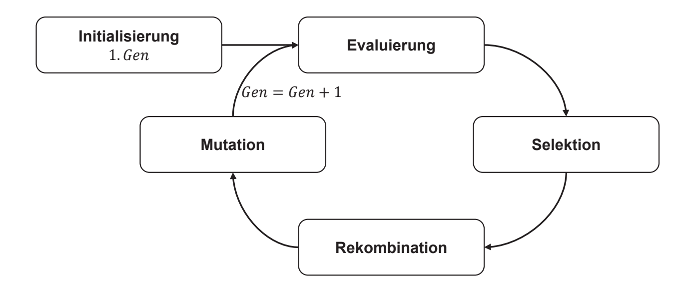


Abbildung 3.2: Iterationsschritte eines genetischen Algorithmus [76]

Zuerst wird eine Population an Individuen initialisiert. Die Individuen repräsentieren einzelne Lösungen des Optimierungsproblems. Bei der Lösung des FJSSPs (vgl. Abschnitt 2.3.1) entspricht ein Individuum dem kompletten Fertigungsablauf, also der vollständigen Abarbeitung aller Bauteile. Da es für die Abarbeitung der Bauteile viele Varianten gibt, z. B. weil Bauteile an alternativen Maschinen gefertigt werden oder die Reihenfolge der Bauteile in einer Warteschlange verändert wird, gibt es viele Individuen. Diese Individuen werden zu einer Population der *1.Generation* zusammengefasst. Im nächsten Schritt werden die Individuen evaluiert. Dafür ermittelt eine sogenannte ‚Fitness-Function' die Lösungsqualität der einzelnen Individuen. [114]

Im Falle von Terminierungsproblemen wird in der Regel die Makespan optimiert, weshalb die Individuen anhand ihrer Makespan verglichen werden. Im darauffolgenden Schritt, der Selektion, werden schlechte Individuen verworfen und Individuen mit einer guten Makespan kopiert. Ziel ist es hierbei, den guten Lösungen eine stärkere Gewichtung zu geben. Um neue Individuen mit einer potenziell noch besseren Makespan-Optimierung zu erhalten, werden die verbleibenden Individuen rekombiniert. Dabei werden Teile der vorhandenen Individuen zu neuen Individuen zusammengesetzt. Zusätzlich erfolgt im darauffolgenden Schritt eine Mutation der Individuen. Dabei werden Teile der Individuen zufällig mutiert, also verändert, um neue Individuen zu erzeugen. Die Population nach der Mutation unterscheidet sich grundlegend von der Population der *1.Generation* und wird

deshalb als *2.Generation* bezeichnet. Diese Iterationsschleife wird so lange ausgeführt, bis entweder eine festgelegte Anzahl an Generationen durchlaufen ist oder die Ergebnisse der Evaluierung eine definierte Schwellwertgrenze überschreiten. Das beste Individuum der letzten Generation erfüllt das Optimierungsziel am besten und kann im Falle der Terminierung als derjenige Fertigungsablauf verstanden werden, der befolgt werden soll. Für die Selektion, die Rekombination und die Mutation existieren verschiedene Strategien, auf die hier nicht weiter im Detail eingegangen wird. [59] [114]

Genetische Algorithmen haben als Hyperheuristik erfolgreich unterschiedliche Terminierungsprobleme gelöst [24]. Eine Hyperheuristik wählt dabei aufgrund der aktuellen Situation eine Heuristik aus, die das vorliegende Optimierungsproblem am besten löst. Diese Hyperheuristik selektiert dabei nicht nur dynamisch eine Heuristik für die gesamte Fertigung, sondern für jede Maschine einzeln aus [95]. Des Weiteren werden verschiedene Terminierungsprobleme, wie das JSSP oder das OSSP, mit GAs gelöst [104] [107]. GAs können die Terminierungsprobleme teilweise optimal lösen, benötigen dafür aber mehrere Stunden Berechnungszeit. Zusätzlich hängt die Lösungsqualität stark von den initialen Parametern und der Problemkomplexität ab, also der Anzahl an Maschinen und Bauteilen. So verfünffacht sich z. B. die Berechnungszeit eines Environments mit 50 Bauteilen und 30 Maschinen im Vergleich zu einem Environment mit 40 Bauteilen und 25 Maschinen [107]. Aufgrund der teilweise sehr langen, und abhängig vom Environment stark schwankenden, Berechnungszeiten ist ein GA nur bedingt für die dynamische Terminierung in einer flexiblen Fertigung einsetzbar.

3.2.4 Reinforcement Learning

Für die Lösung des JSSPs mit Hilfe von Reinforcement-Learning-Methoden (vgl. Abschnitt 2.4) werden verschiedene Ansätze untersucht. Dabei unterscheiden sich die Methoden grundlegend im Aufbau der Agenten. Während in einigen Publikationen Single-Agenten (vgl. Abschnitt 2.4.4) genutzt werden, setzen andere Veröffentlichungen Multi-Agenten Systeme (vgl. Abschnitt 2.4.4) ein.

Das JSSP kann erfolgreich mit einem Single-Agent gelöst werden, der ein Deep Q-Network (vgl. Abschnitt 2.4.5) nutzt [130]. In der diesbezüglich veröffentlichten Literatur werden die Ergebnisse des entwickelten Algorithmus mit Heuristiken, u. a. der SPT-Heuristik, in einem Environment (vgl. Kapitel 2.4.4) mit 5 Maschinen und 10 FAUFs verglichen [130]. Dabei stellt sich heraus, dass die Terminierungsergebnisse der vorgestellten Methode nicht besser sind als diejenigen der zum Vergleich herangezogenen Heuristiken. Die Autoren weisen aber darauf hin, dass sie grundsätzlich die Nutzbarkeit eines DQNs zum Lösen des JSSPs zeigen [130]. Auf die Frage, welche Möglichkeiten es für qualitative Verbesserungen des Terminierungsergebnisses gibt, wird nicht weiter eingegangen. Derselbe Ansatz wird auch in anderen Publikationen genutzt, ohne zusätzliche Erkenntnisse zu liefern [142] [58].

Ein weiterer Ansatz, das JSSP mittels RL zu lösen, basiert auf der Policy-Gradient-Methode [128]. Diese Methode beruht auf einem Policy-based-Algorithmus (vgl. Anhang A.1). Die Ergebnisse der in der Veröffentlichung [128] vorgestellten Methode werden mit Ergebnissen von Heuristiken und kommerziellen Solvern, also exakten Lösungsmethoden, verglichen. Bei der Evaluierung der Ergebnisse zeigt sich, dass die präsentierte Methode in jedem Fall besser als die gegenübergestellten Heuristiken ist, aber in jedem Fall $\approx$10 % schlechter als die exakten Lösungsmethoden.

Die Autoren weisen zusätzlich darauf hin, dass auch längere Trainingszeiten nicht zu besseren Ergebnissen führen, da der Algorithmus ein lokales Optimum findet und aus diesem nicht mehr herauskommt. Das lässt darauf schließen, dass die vorgestellte Methode nicht sehr robust ist. Das Ergebnis nach dem Training ist abhängig von der Qualität des lokalen Optimums - es sind somit mehrere Trainingsläufe notwendig, in denen verschiedene lokale Optima gefunden werden, um zu einem guten Terminierungsergebnis zu gelangen. Wie viele unterschiedliche Trainingsläufe notwendig sind, um auf die präsentierten Ergebnisse zu kommen, wird nicht beschrieben. Wird die Komplexität des Problems erhöht, weil z. B. alternative Maschinen verwendet werden, Bauteile verschiedene Prioritäten aufweisen oder Störungen auftreten, vergrößert sich der Lösungsraum exponentiell. Das hat zur Folge, dass es auch mehr lokale Optima geben kann. In diesen Fällen

nimmt die Robustheit der vorgestellten Methode weiter ab. Aus diesem Grund eignet sich diese Methode in der vorgestellten Form nicht für die dynamische Terminierung.

Andere Publikationen (z. B. [130]) schlagen Multi-Agenten-Methoden vor, die weder Bauteile noch Maschinen direkt steuern, sondern aus einem Pool aus vorher definierten Heuristiken jene ausgeben, die für die aktuelle Situation am besten geeignet ist. Dabei wird jede Maschine durch ihren eigenen Agenten repräsentiert, der eine Heuristik auswählt, anhand derer ein Bauteil aus der Warteschlange ausgewählt wird, das als nächstes gefertigt werden soll. Diese vorgestellte Methode, nach [130], besitzt zwei wesentliche Nachteile:

- Die Methode greift erst ein, wenn die Bauteile bereits einer Maschine zugeordnet sind. Ein Umleiten auf eine andere Maschine ist ab diesem Zeitpunkt nicht mehr möglich. Diese Annahme vereinfacht das Optimierungsproblem stark und ist beim JSSP zulässig, da es keine alternativen Maschinen gibt. In der Fertigung wird hier von einer Reihenfertigung gesprochen (vgl. Abschnitt 2.1.2). Bei Fertigungen mit alternativen Bearbeitungsmaschinen, z. B. der Matrixfertigung, ist diese Annahme jedoch nicht zulässig. Daher eignet sich diese Methode nicht für die Terminierung von Fertigungen mit alternativen Bearbeitungsmaschinen.
- Das Optimierungspotenzial beschränkt sich auf Heuristiken, die vor der Terminierung definiert werden. Dabei können allgemeingültige Heuristiken wie FIFO oder SPT zur Anwendung kommen. Um das Terminierungsergebnis weiter zu verbessern, können eigene Heuristiken für die entsprechende Fertigung definiert werden, was viel Domänenwissen und einen großen zeitlichen Aufwand benötigt [126]. Die Ergebnisse der betreffenden Veröffentlichung zeigen, wie nahe eine allgemeingültige Heuristik an die Terminierungsergebnisse dieser Methode herankommt. Der Aufwand, einen solchen Multi-Agenten-Ansatz in der Fertigung zu implementieren, steht daher in keinem Verhältnis zum erzielten Terminierungsergebnis.

Andere Publikationen entwickeln RL-Methoden, die den verschiedenen Bauteilen zu jedem Zeitschritt vorgeben, an welcher Maschine sie optimalerweise gefertigt werden sollen [13] [82]. Dabei wird in der Regel nicht auf einen Single-Agent, sondern auf einen Multi-Agenten-Ansatz zurückgegriffen. Grund dafür ist die notwendige Skalierbarkeit der Lösung. Ein Single-Agent ist auf eine feste Anzahl von Maschinen und Bauteilen trainiert, wohingegen bei einer Multi-Agenten-Methode weitere Agenten hinzugefügt bzw. entfernt werden können. Ein weiterer Grund für den Einsatz von Multi-Agenten-Systemen ist die Lern-Performance. Bei einem Single-Agent werden der State- und der Action-Vektor (vgl. Abschnitt 2.4.4) mit zunehmender Anzahl von Maschinen und Bauteilen exponentiell größer. Mit wachsender Größe dieser beiden Vektoren nimmt die Lern-Performance ab, bis der Single-Agent ab einer bestimmten Netzgröße gar nicht mehr lernen kann [45].

Aus diesem Grund wurde ein Multi-Agent Deep Q-Network (vgl. Abschnitt 2.4.5) für die Terminierung entwickelt [13]. In der diesbezüglichen Publikation besitzt jedes Bauteil seinen eigenen Agenten. Die Agenten kommunizieren dabei nicht untereinander, sondern erhalten alle Informationen über den State-Vektor aus dem Environment. Evaluiert wird die Methode an einem FJSSP (vgl. Abschnitt 2.3.1). *(Anmerkung: In der Veröffentlichung selbst wird von einem JSSP gesprochen. Im Verlauf des Textes wird aber festgelegt, dass es alternative Maschinen gibt, was es zu einem FJSSP macht.)* Das evaluierte Environment ist mit sechs Maschinen und drei Bauteilen so klein, dass sich keine Warteschlangen bilden können. Zusätzlich können sich die Produkte für jeden Bearbeitungsschritt zwischen zwei Maschinen entscheiden. Das führt dazu, dass die Maschinen die meiste Zeit ungenutzt sind und auf Produkte warten. Sinnvollerweise sollten Evaluierungen für die Produktionsterminierung in Environments stattfinden, bei denen es mehr Produkte als Maschinen gibt, damit das Verhalten der Produkte in Warteschlangen bzw. im Falle von Konflikten bei der Maschinenbelegung untersucht werden kann.

Andere Publikationen greifen auf eine weitere RL-Methode für den Multi-Agenten-Ansatz zurück. So wird beispielsweise gezeigt, dass auch Multi-Agent Policy Gradient-Methoden das JSSP lösen können [82]. Aufbau und Funktionsweise der Multi-Agent-Proximal-Policy-Optimization (MAPPO) sind im Anhang beschrieben (vgl. Anhang A.1). In der betreffenden Veröffentlichung wird die

Leistungsfähigkeit der entwickelten Methode in größeren Environments mit bis zu vier Maschinen und zehn Bauteilen untersucht. Zusätzlich wird das Ergebnis des Algorithmus mit einer selbst entwickelten Heuristik verglichen. Die MAPPO zeigt in den verschiedenen Environments bessere Ergebnisse als die gegenübergestellte Heuristik, jedoch fehlt ein Vergleich mit einem optimalen Ergebnis. Grundsätzlich lassen sich für beide Ansätze, [13] und [82], folgende Probleme identifizieren:

- Da die Bauteile in beiden Fällen nicht miteinander kommunizieren und die verschiedenen States auswendig lernen, findet ein sogenanntes ‚overfitting' statt. Beim Training werden die gleichen States so oft durchlaufen, bis die Agenten die beste Policy gefunden haben. Die Agenten erzielen aber nur in denjenigen States gute Terminierungsergebnisse, die sie vorher trainiert haben. Eine Generalisierung des Wissens auf States, die den gelernten States ähnlich sind, findet nicht statt. Besonders in einem Fertigungsumfeld mit vielen Maschinen und Bauteilen hat das zur Folge, dass beim Training alle möglichen Szenarien mehrmals durchlaufen werden müssen - was zu extrem langen Trainingszeiten führt.
- Auf die Reihenfolge, in der die Agenten ihre Entscheidung treffen, wird nicht eingegangen. Besitzen Bauteile beispielsweise unterschiedliche Prioritäten und die Bauteil-Agenten mit den niedrigen Prioritäten entscheiden als Erstes, welches Bauteil zu welcher Maschine gehen soll, führt dies dazu, dass das Optimierungsproblem eventuell nicht optimal für das nachfolgende Bauteil mit einer höheren Priorität gelöst wird [82]. Beide Veröffentlichungen lösen dieses Problem nicht.

3.2.5 MCTS kombiniert mit Reinforcement Learning

Die Kombinationsmöglichkeiten eines Tree-Search-Algorithmus, wie der MCTS, mit RL werden in Abschnitt 2.5.4 beschrieben. Erforscht und umgesetzt wurde diese Kombination zunächst erfolgreich am Brettspiel ‚Go' [120]. Bei einer Brettgröße von $19 \cdot 19$ Feldern gibt es $2.1 \cdot 10^{170}$ erlaubte Spielzüge - die Komplexität des

Problems ist damit sehr groß. Um den großen Lösungsraum zielgerichtet durchsuchen zu können, wird die MCTS dabei von einem NN unterstützt (vgl. Abschnitt 2.5.4). Die entwickelte Methode, um MCTS und RL zu kombinieren, wurde daraufhin zum Lösen von anderen Optimierungsproblemen, wie dem JSSP genutzt [110]. In der betreffenden Veröffentlichung wird eine Terminierungs-Methode für die Blechbearbeitungsproduktion vorgestellt. Die Bleche durchlaufen dabei drei Stationen: den Blechzuschnitt, das Biegen und die Montage, wobei das Biegen von manchen Bauteilen übersprungen werden kann. Das Terminierungsproblem wird als JSSP beschrieben, wobei die Komplexität des Problems auf grund von Beschränkungen beim Blechzuschnitt und der Montage verringert wird. Die Rollouts der MCTS (vgl. Abschnitt 2.5.2) werden durch ein Single-Agent DQN unterstützt. Anstatt des Durchlaufens vieler Monte-Carlo-Evaluationen, um die Qualität eines States zu ermitteln, übergibt hier ein antrainiertes NN den Knotenwert (vgl. Abschnitt 2.5.4). Diese Methode besitzt folgende Nachteile:

- Die Methode dient zur Lösung eines JSSPs. Die Größe des Lösungsraumes des Terminierungsproblems wird durch verschiedene Beschränkungen weiter verringert, sodass die Komplexität des evaluierten Problems nicht abschätzbar ist. Da ein JSSP angenommen wird, gibt es keine parallelen Maschinen, sodass dieser Ansatz nicht in einer Matrixfertigung angewendet werden kann.
- Die Trainingszeit des NN beträgt fünf Tage. Eine schnelle Reaktion auf unvorhergesehene Störungen ist damit nicht möglich. Die Autoren weisen selbst darauf hin, dass die präsentierte Methode einen APS-Run ablösen kann, aber nicht für die schnelle Reaktion in der Fertigung geeignet ist. Ziel ist es hierbei vielmehr, eine initiale Terminierung zu erzeugen.

- Die Terminierungsergebnisse der vorgestellten Methode werden mit einer Heuristik verglichen. Die verwendete Heuristik ist nicht auf den Anwendungsfall spezialisiert und fertigt als Erstes die Bauteile mit einem früheren Liefertermin. Sowohl die Heuristik als auch die entwickelte Methode erzielen sehr ähnliche Durchlaufzeiten für die Bauteile. Die Heuristik löst das Problem innerhalb weniger Sekunden und muss nicht angelernt werden. Da solche klassischen Heuristiken Optimierungsprobleme in der Regel nicht optimal lösen können, ist auch hier anzunehmen, dass das Problem nicht optimal gelöst wird. Einen Vergleich zu einer optimalen Lösung liefert die Veröffentlichung nicht.

3.3 Anforderungen an die dynamische Terminierung für eine flexible Fertigung

Die mathematische Lösung von Terminierungsproblemen wird vielseitig untersucht und viele Ergebnisse dazu sind publiziert. Dabei werden verschiedene Terminierungsprobleme definiert und mit unterschiedlichen Methoden gelöst (vgl. Abschnitt 3.2). Dennoch gibt es weiteren Forschungsbedarf, da nicht alle Terminierungsprobleme hinsichtlich der Lösungsqualität und der Berechnungszeit ausreichend gelöst wurden. Untersuchungen von Terminierungsproblemen beschränken sich in der Regel auf das JSSP [103]. Die hinreichende Lösung von Terminierungsproblemen für Fertigungen mit größeren Freiheitsgraden, wie z. B. dem FJSSP oder dem OSSP, steht noch aus. Auf Grundlage der im Abschnitt zum Stand der Forschung und Industrie untersuchten Publikationen können mehrere Anforderungen an eine dynamische Terminierung in einer flexiblen Fertigung identifiziert werden. Diese Anforderungen dienen als Grundlage, um die Forschungslücke aufzuzeigen und die nachfolgend entwickelte Methode zu evaluieren.

- **Komplexität**: Die Methode für die dynamische Terminierung muss dazu in der Lage sein, alternative Maschinen, Arbeitsplätze und Werkzeuge in die Terminierung miteinzubeziehen. Die Komplexität des Terminierungsproblems wird damit sehr groß, viel größer als das JSSP.
- **Robustheit**: Die Lösungsqualität von Optimierungsmethoden, die eine Lösung approximieren, unterliegt Schwankungen. Damit die Terminierungsergebnisse der dynamischen Terminierung in der Fertigung genutzt werden können, müssen diese robust sein. Unter Robustheit wird in diesem Zusammenhang verstanden, dass die Ergebnisse mehrerer Terminierungsläufe möglichst nah zusammen liegen und Ausreißer, z. B. im Falle eines lokalen Optimums, ausgeschlossen werden können.
- **Action-Combination**: Wird ein Multi-Agenten-Ansatz verwendet, muss sichergestellt werden, dass FAUF-Agenten bei der Auswahl der nächsten Fertigungsstation nicht nur sich, sondern auch die anderen FAUF-Agenten miteinbeziehen. Geschieht das nicht, kann das Terminierungsproblem nicht optimal gelöst werden [82]. Die Kombination der Actions der verschiedenen Produkt-Agenten wird als Action-Combination-Problem definiert.
- **Generalisierungsfähigkeit**: Da nicht alle möglichen Szenarien und Störungen vor der Fertigung bekannt sind bzw. simuliert oder trainiert werden können, muss eine RL-Methode eine starke Generalisierungsfähigkeit besitzen (vgl. Abschnitt 2.4.4), um auch in unbekannten Fällen gute Terminierungsergebnisse sicherzustellen.
- **Reaktionszeit**: Im Falle einer Störung in der Fertigung muss das Ergebnis einer dynamischen Terminierung innerhalb weniger Minuten vorliegen. Die genaue Reaktionszeit orientiert sich an den Durchlauf- und Prozesszeiten der Fertigung und kann deshalb nicht allgemeingültig festgelegt werden. Trotzdem ist davon auszugehen, dass ein Ergebnis von unter 5 Minuten für die meisten Fertigungen ausreichend ist.

3.4 Fazit aus den Analysen zum Stand der Forschung

Für die Lösung von Terminierungsproblemen in der Auftragseinplanung gibt es zahlreiche IT-System-Anbieter (vgl. Tabelle 3.2). Die vorhandenen IT-Systeme sind in der Lage, sowohl lang- und mittelfristige Planungshorizonte zu betrachten als auch die Fertigungsterminierung zu übernehmen. Je nach Komplexität des Terminierungsproblems dauert ein Terminierungslauf bis zu mehreren Stunden (vgl. Abschnitt 2.2.1). Tritt eine Störung auf, kann dem geplanten Fertigungsablauf nicht mehr gefolgt werden. Da eine Neuterminierung eine lange Berechnungszeit benötigt, werden die FAUFs von einem Steuerer umpriorisiert und neu auf die Maschinen/Arbeitsplätze verteilt (vgl. Abschnitt 2.2.1). Diese Eingriffe beruhen auf dem Erfahrungswissen des Steuerers und berücksichtigen die Auswirkungen der Eingriffe auf nachfolgende Fertigungsstationen nur sehr bedingt - eine optimale Fertigungsauslastung wird so nicht erreicht. Aus Fertigungssicht ist ein IT-System erforderlich, das eine ereignisgesteuerte, dynamische Terminierung ermöglicht. Im Falle einer Störung werden die FAUFs so umterminiert, dass ein möglichst optimaler Fertigungsablauf gewährleistet wird. Dabei optimiert dieses IT-System die Terminierung aller FAUFs über die gesamte Fertigung. Die Reaktionszeit eines solchen IT-Systemes ist mit ausschlaggebend für das Optimierungspotenzial und somit möglichst gering zu halten - Ziel sollte eine Reaktionszeit von wenigen Minuten sein. Die Anforderungen an eine dynamische Terminierung einer flexiblen Fertigung werden in Abschnitt 3.3 hergeleitet.

Unterschiedliche wissenschaftliche Methoden stehen zur Verfügung, um die Terminierung komplexer Fertigungsszenarien zu ermöglichen und zu beschleunigen. Auf der wissenschaftlichen Seite beschäftigen sich seit Jahren Forschungsansätze mit (Meta-)Heuristiken und genetischen Algorithmen (vgl. Abschnitte 3.2.2 und 3.2.3). Die Entwicklung neuer RL-Methoden schafft die Voraussetzungen für die Lösung verschiedener komplexer Terminierungsprobleme. Für die Lösung des JSSPs sind Single- und Multi-Agent-Ansätze veröffentlicht. Zusätzlich werden verschiedene RL-Methoden kombiniert, um das Terminierungsproblem effizienter zu lösen. Entsprechend den Anforderungen aus Abschnitt 3.3 werden

nachfolgend die in Abschnitt 3.2 vorgestellten RL-Methoden und Veröffentlichungen eingeordnet und die Forschungslücke wird identifiziert (vgl. Tabelle 3.3). Wird eine Anforderung voll erfüllt, wird das mit einem + gekennzeichnet. Teilweise erfüllte Anforderungen werden mit einem (+) kenntlich gemacht. VA steht für ‚vorliegende Arbeit'.

Tabelle 3.3: Analyse der Forschungslücke in den relevanten Veröffentlichungen

	Merkmale	[130]	[128]	[80]	[13]	[82]	[110]	[58]	VA
RL-Methode	Single-Agent RL	+	+					+	
	Multi-Agent RL			+	+	+			
	MCTS+SA						+		
	MCTS+MA								+
Anforderungen	Komplexität - OSSP								+
	Robustheit								+
	Action-Combination	(+)	(+)				(+)		+
	Generalisierungsfähigkeit		(+)	+	+	(+)		(+)	+
	Dynamische Terminierung	(+)	(+)	(+)	+	+			+

Anmerkung: VA - Vorliegende Arbeit

In der Forschung findet das JSSP die meiste Beachtung. Das Problem kann bereits mit unterschiedlichen Methoden exakt bzw. nahezu optimal gelöst werden. Da dieses Terminierungsproblem jedoch keine alternativen Maschine/Arbeitsplätze beinhaltet, können die dafür entwickelten Methoden nicht in flexiblen Fertigungen eingesetzt werden, die alternative Maschinen/Arbeitsplätze besitzen. Der Lösungsraum wird in diesem Fall exponentiell größer und damit wird das Terminierungsproblem schwerer lösbar. Für die Auftragsterminierung einer flexiblen Fertigung, mit alternativen Maschinen/Arbeitsplätzen, muss mindestens das FJSSP oder das OSSP gelöst werden können. Keine der vorgestellten Veröffentlichungen löst Probleme mit dieser Komplexität.

Das Ergebnis der dynamischen Terminierung muss robust sein. Darunter wird verstanden, dass die Ergebnisse reproduzierbar sein müssen und innerhalb eines gewissen Konfidenzintervalls liegen. Die betrachteten Publikationen gehen auf diese Anforderung nicht ein. Das Problem der Robustheit wird zwar erwähnt, aber nicht gelöst [128].

Wird ein Multi-Agenten-Ansatz verwendet, müssen die Actions der einzelnen Agenten so kombiniert werden, dass ein globales Optimum erzeugt wird. Die einzelnen Agenten optimieren nur ihre eigenen Actions, ohne die anderen Agenten in die Action-Auswahl miteinzubeziehen. Von den analysierten Veröffentlichungen, die einen Multi-Agenten-Ansatz nutzen, wird bei keiner eine Lösung des Action-Combination-Problems beschrieben. Bei den Publikationen, die einen Single-Agent verwenden, ist die Anforderung teilweise erfüllt, da ein Single-Agent automatisch alle Bauteile betrachtet und keine separate Action-Combination notwendig ist.

Die Generalisierungsfähigkeit der Methode zur dynamischen Terminierung ist eine zwingende Voraussetzung, da nicht alle möglichen Störungsszenarien im Vorfeld berechnet und terminiert werden können. Die beim Training erlernten Zusammenhänge müssen auf unbekannte Situationen anwendbar sein. Die Generalisierungsfähigkeit von RL ist ein Grund dafür, warum es sinnvoll erscheint, diese Methode für die dynamische Terminierung einzusetzen. Verschiedene Veröffentlichungen sind auf diese Anforderungen eingegangen. Die Publikation eines umfänglichen Frameworks für das Training zur Maximierung der Generalisierungsfähigkeit ist jedoch in keinem der analysierten Literaturbeiträge zu finden.

Die meisten untersuchten Arbeiten beschäftigen sich mit der initialen Terminierung einer Fertigung und nicht mit der dynamischen Terminierung, bei der die neue Terminierung in wenigen Minuten bereitstehen muss.

Als Fazit zu Tabelle 3.3 ist zusammenfassend festzustellen, dass aktuell keine Methode bekannt ist, die alle Anforderungen einer flexiblen Fertigung an die dynamische Terminierung erfüllt. Besonders komplexe Terminierungsprobleme finden aufgrund der enormen Größe des Lösungsraums wenig Beachtung in der

Forschung. Zusätzlich ist die Robustheit für eine Anwendung in der realen Fertigung unabdingbar. Die Tatsache, dass in den vorgestellten Veröffentlichungen nicht auf die Robustheit der Lösungen eingegangen wird, lässt auf die eher akademische Natur dieser Ansätze schließen, die dementsprechend eine grundsätzliche Machbarkeit untersuchen.

4 Methode zum Einsatz von RL für die dynamische Fertigungsdurchlaufsteuerung

In diesem Kapitel wird die Methode zur dynamischen Terminierung entwickelt. Grundlagen für die Methode sind die Anforderungen einer flexiblen Fertigung aus Abschnitt 3.3. Nach einer Präzisierung der gegebenen Anforderungen und Voraussetzungen in Abschnitt 4.1 erfolgt die detaillierte Konzipierung der Fertigungsdurchlaufsteuerung mit RL in Abschnitt 4.2.

4.1 Anforderungen und Voraussetzungen für die entwickelte Methode

Im vorliegenden Abschnitt werden zum einen die Anforderungen an den Fertigungsprozess als auch die Anforderungen an die Programmarchitektur definiert. Zusätzlich wird das Terminierungsproblem und das Optimierungsziel beschrieben.

4.1.1 Anforderungen an die Programmarchitektur

Da in der zu entwickelnden Methode für die dynamische Terminierung unterschiedliche RL-Methoden kombiniert werden, ist es sinnvoll, dass die Programmarchitektur aus verschiedenen Funktionsmodulen besteht. Damit folgt die

Programmarchitektur der vorgestellten Methode dem ‚Liskovsches Substitutionsprinzip' der objektorientierten Programmierung und ist dementsprechend logisch, nachvollziehbar, erweiterbar sowie änderbar [89]. Die einzelnen Funktionsmodule können ausgetauscht werden und kommunizieren nur über Schnittstellen.

Funktionsmodule umfassen dabei sowohl einzelne RL-Methoden als auch das sogenannte Environment (vgl. Abschnitt 2.4.4), also die Fertigung inklusive aller fertigungsrelevanten Zusammenhänge wie der Anzahl an FAUFs und der Maschinen, Arbeitspläne oder Fertigungszeiten. Das Environment kann somit leicht ausgetauscht werden. Da in der Validierung (vgl. Kapitel 5) verschiedene Environments getestet werden, lässt sich so zusätzlich sicherstellen, dass die Validierungsergebnisse nur aufgrund der unterschiedlichen Environments zustande kommen und nicht auf weitere Änderungen anderer Funktionsmodule zurückzuführen sind.

4.1.2 Anforderungen an den Fertigungsprozess

Während der kurzfristigen Planung bzw. Terminierung findet ein APS-System eine optimale oder nahezu optimale Lösung für das gegebene Terminierungsproblem (vgl. Abschnitt 3.1). Tritt eine Störung auf, muss der aktuelle Zustand der Fertigung an die dynamische Terminierung übergeben werden. Dieser Factory-State beinhaltet Informationen über den Bearbeitungsstand der FAUFs und die Maschinenverfügbarkeit. Anhand dieser Eingangsgröße ermittelt die dynamische Terminierung als Nächstes die neue Terminierung der Fertigung und gibt diese in Form eines Action-Vektors zurück an die Fertigung. Damit zeitnah eine neue Terminierung erstellt werden kann, müssen statische Informationen sowohl aus dem ERP- als auch aus dem APS-System vorliegen. Zu den statischen Informationen zählen:

- Arbeitspläne inklusive alternativer Arbeitsfolgen, Arbeitsplätze oder Arbeitsvorgänge
- eingeplante FAUFs
- ablaufoptimierte Terminierung aus dem APS-System

- Ressourcenverfügbarkeit
- Mitarbeiterqualifikationen
- Schichtpläne bzw. Mitarbeiterverfügbarkeit
- Fertigungs-, Rüst- und Transportzeiten

Diese statischen Informationen werden durch dynamische Informationen aus der Fertigung ergänzt. Üblicherweise werden die letztere von einem MES verwaltet. Zu den dynamischen Informationen gehören:

- Maschinenstatus (gestört, belegt, frei)
- Mitarbeiterkapazitäten
- Rückmeldungen der einzelnen Arbeitsvorgänge der aktuellen FAUFs
- nicht eingeplante Eilaufträge

Die Anfrage zur Erstellung einer neuen dynamischen Terminierung kann ereignisbasiert oder kontinuierlich erfolgen. Die ereignisbasierte dynamische Terminierung wird angestoßen, wenn ein Ereignis bzw. eine Störung auftritt. Bis zum Störungszeitpunkt verläuft die Fertigung wie in der Terminierung des APS-Systems berechnet. Bei Auftritt der Störung ermittelt die dynamische Terminierung eine neue Terminierung, die die aktuelle Situation miteinbezieht. Der dynamisch erstellten Terminierung wird ab diesem Zeitpunkt im Fertigungsablauf bis zur nächsten Störung oder dem Schichtende gefolgt. Im Anschluss daran erzeugt ein APS-System eine neue Terminierung für die nächste Schicht oder den nächsten Tag. Bei Fertigungen mit wenigen Störungen und großen Pufferzeiten kann diese Art der dynamischen Terminierung angewendet werden. Normale Abweichungen im Fertigungsprozess und kleinere Störungen können über die Pufferzeiten abgefangen werden und wirken sich nicht negativ auf die Fertigung aus. Die dynamische Terminierung greift nur bei Störungen ab einer vordefinierten Zeitüberschreitung bzw. einem Schwellwert ein.

Im Gegensatz dazu wird bei der kontinuierlichen Terminierung nach einer festgelegten Zeit automatisch eine neue Terminierung erzeugt. Die FAUFs folgen der aktiven Terminierung so lange, bis eine neue Terminierung bereit steht. In einer Fertigung mit geringen Puffern und sich stark voneinander unterscheidenden Prozesszeiten erscheint diese Terminierung sinnvoll. Maschinenstörungen und Verzögerungen im Fertigungsprozess addieren sich so nicht auf.

In Abschnitt 3.3 werden die Anforderungen an die dynamische Terminierung einer flexiblen Fertigung definiert. Damit die dynamische Terminierung in einer Fertigung genutzt werden kann, muss die Methode mindestens das FJSSP (vgl. Abschnitt 2.3.1) lösen können, da es in der Regel alternative Maschinen/Arbeitsplätze gibt. Es muss zusätzlich sichergestellt werden, dass das Terminierungsergebnis mit einer großen Wahrscheinlichkeit zu einem nahezu optimalen Fertigungsablauf führt. Statistische Ausreißer und lokale Optima müssen daher vermieden bzw. ausgeglichen werden. Die zu entwickelnde Methode muss auf Fertigungsszenarien anwendbar sein, die im Vorfeld unbekannt sind, und trotzdem eine nahezu optimale Terminierung berechnen. Die dynamische Terminierung lässt sich aufgrund des Planungshorizontes und der Reaktionszeit von der klassischen Terminierung unterscheiden. Für die **dynamische Terminierung** wird in der vorliegenden Arbeit eine Reaktionszeit von **unter 3 Minuten** definiert.

4.1.3 Definition des Optimierungsziels

Wie in Abschnitt 2.3.2 dargestellt, gibt es für die Terminierung mehrere Optimierungsziele. Dabei wird in wissenschaftlichen Publikationen, die sich mit der Lösung des FJSSPs oder des OSSPs befassen, in der Regel eine zeitliche Optimierung untersucht. Für die zeitliche Optimierung des vorliegenden Terminierungsproblems erscheinen grundsätzlich sowohl die Makespan als auch die Total-Lead-Time-Optimierung sinnvoll. Beide Optimierungsziele minimieren die Fertigungszeit, allerdings setzt sich Letztere bei beiden Zielen anders zusammen. Bei der Makespan-Optimierung wird die Zeit von Beginn bis zur Fertigstellung aller Fertigungsaufträge minimiert. Die Fertigungszeit hängt damit allein von der

Fertigstellung des letzten FAUFs ab. Bei der Total-Lead-Time-Optimierung werden die einzelnen Fertigungszeiten der FAUFs aufsummiert. Somit hat die Fertigungszeit jedes einzelnen FAUFs Auswirkungen auf die Gesamtfertigungszeit und nicht nur die Fertigstellungszeit des letzten FAUFs. Beide Optimierungsziele können FAUF-Prioritäten mittels Gewichtungen abbilden (vgl. Abschnitt 2.3.2).

Aufgrund des größeren Einflusses der einzelnen Fertigungszeiten wird die Total Lead Time als Optimierungsziel der nachfolgenden Methode und ihrer Validierung festgelegt (vgl. Abschnitt 2.3.2). Die Reward-Funktion (vgl. Abschnitt 2.4.4) wird deshalb so gewählt, dass die KI-Methode die Total Lead Time minimiert. An dieser Stelle wird darauf hingewiesen, dass die Terminierung einer realen Fertigung nicht nur die Fertigungszeit, sondern auch die Fertigungskosten und andere Optimierungsziele berücksichtigt. Diese Anpassbarkeit des Optimierungsziels ist somit eine weitere wesentliche Anforderung an die zu entwickelnde Methode.

4.1.4 Annahmen für die Lösung des Terminierungsproblems

Nachfolgend wird eine Methode für die dynamische Terminierung entwickelt, die folgende Annahmen voraussetzt:

- Ein OSSP mit einem Set aus $\mathcal{K}$ Maschinen $\mathcal{K} = \{1, \ldots, K\}$ und N FAUFs $\mathcal{P} = \{1, \ldots, N\}$ wird gelöst.
- Unterschiedliche Bearbeitungs-, Transport- und Rüstzeiten werden betrachtet.
- Die FAUFs können unterschiedliche Prioritäten besitzen.
- Die FAUFs können entweder zu einer Maschine transportiert werden oder in ihrer aktuellen Position warten.
- Während des Fertigungsprozesses an einer Maschine kann ein FAUF die Maschine nicht verlassen.

- Störungen können in jedem Zeitschritt auftreten. Ein FAUF, der sich zum Eintrittszeitpunkt der Störung in der entsprechenden Bearbeitungsmaschine befindet, bleibt bis zur Behebung der Störung in der Maschine. FAUFs in der Warteschlange können diese verlassen und andere Maschinen anfahren.
- Auf jeder Maschine kann nur ein FAUF gleichzeitig gefertigt werden. Daraus ergibt sich eine Fallunterscheidung:
  - Maschinen mit Warteschlangen: Mehrere FAUFs können gleichzeitig zu diesen Maschinen transportiert werden. Die Auswahl des als Nächstes zu fertigenden FAUFs kann z. B. nach der FIFO-Heuristik, Prioritäten, o. Ä. erfolgen.
  - Maschinen ohne Warteschlangen: Hierbei darf nur ein FAUF zu einer Maschine transportiert werden. Andere FAUFs, die im selben Zeitschritt zu dieser Maschine wollen, müssen warten oder eine andere Maschine für den AVO wählen.

4.2 Fertigungsdurchlaufsteuerung mit RL

In diesem Abschnitt wird eine Methode für die Prozessdurchlaufsteuerung einer flexiblen Fertigung mittels einer **dynamischen Terminierung** entwickelt. Dafür wird zunächst auf die Prozessdurchlaufsteuerung an sich eingegangen, bevor die eigentliche Methode konzipiert wird.

4.2.1 Systemarchitektur MCTS und MADQN

Nachfolgend wird die ausgearbeitete Systemarchitektur vorgestellt, die es ermöglicht, die MCTS (vgl. Abschnitt 2.5.2) zusammen mit einem Multi-Agent-DQN (MADQN) (vgl. Abschnitt 2.4.5) für die dynamische Terminierung einzusetzen. Die einzelnen Bestandteile der Systemarchitektur sind modular aufgebaut und

lassen sich über Schnittstellen austauschen. Das Unified-Modeling-Language-Klassendiagramm in Abbildung 4.1 zeigt die Zusammenhänge der einzelnen Objekte und deren zugrundeliegende Funktionen.

Die einzelnen Klassen in Abbildung 4.1 sind modular aufgebaut und kommunizieren über Schnittstellen (en: interfaces) miteinander. Einzelne Klassen wie die des **MADQN** können so leicht ausgetauscht werden. Das dargestellte Klassendiagramm beinhaltet vier Komponenten: die **Factory**, also die Fertigung, die **MCTS** (vgl. Abschnitt 4.2.2), das **MADQN** (vgl. Abschnitt 4.2.3) und die Klasse für das Clustern der FAUFs (vgl. Abschnitt 4.2.5).

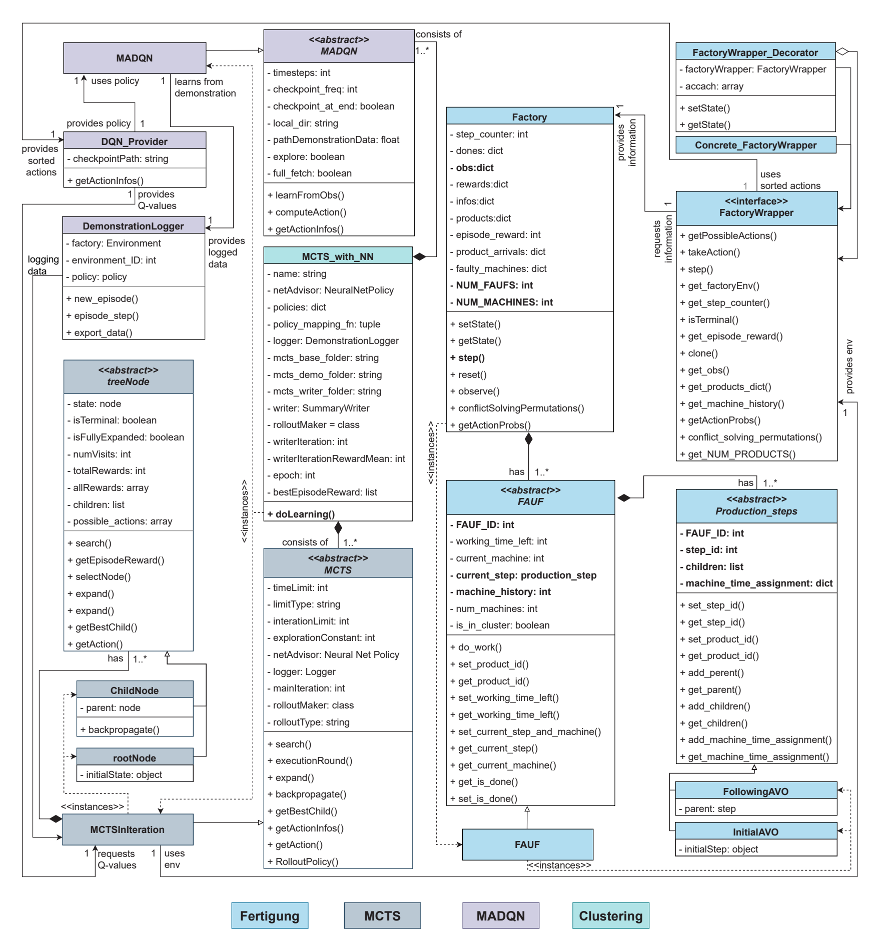


Abbildung 4.1: Unified-Modeling-Language-Klassendiagramm der Methode für die dynamische Terminierung

Wird nun eine dynamische Terminierung angestoßen, wird zunächst die Funktion **doLearning** der Klasse **MCTS_with_NN** ausgeführt. Ausgehend von dieser Funktion wird die **Factory** über den **FactoryWrapper** initialisiert. Die Klasse **Factory** beinhaltet grundsätzliche Informationen über die Fertigung, z. B. über die Anzahl der Maschinen (**NUM_MACHINES**) und FAUFs (**NUM_FAUFS**).

Der **FactoryWrapper** entkoppelt die **Factory** von den anderen Methoden. Dieser Aufbau folgt dem Prinzip der Schnittstellentrennung [96]. Die FAUFs der Klasse **Factory** werden über die abstrakte Klasse **FAUF** initialisiert. Jedem FAUF wird darin eine eindeutige Nummer bzw. ID zugewiesen. Zusätzlich wird über die Variable **machine_history** der Fertigungsstatus des FAUFs in Form der bereits durchlaufenen Maschinen angegeben. Bei der Initialisierung eines FAUFs wird der **production_step** über die abstrakte Klasse **Production_steps** initialisiert. Die abstrakte Klasse **Production_steps** ermöglicht die Abbildung eines Arbeitsplanes (vgl. Abschnitt 2.2.2), da sowohl der aktuelle AVO **(InitialAVO)**, als auch alle nachfolgenden AVOs **(FollowingAVO)** in dieser Klasse enthalten sind. Dieser Aufbau wird ‚Schablonenmethode' genannt, da die abstrakte Klasse durch zwei Unterklassen definiert ist [96].

Ist die **Factory** inklusive aller FAUFs initialisiert, wird über die Klasse **MCTS_with_NN** eine weitere Klasse, die **MCTSInIteration**, initialisiert. Die Klasse **MCTSInIteration** erbt ihre Attribute von der Klasse **MCTS_with_NN**. Diese Architektur dient dem Entkoppeln der einzelnen Methoden und wird als ‚Fabrikmethode' bezeichnet [96]. Um den Zustand der Factory zu synchronisieren, werden alle Funktionen des **FactoryWrapper** mit dem **FactoryWrapper_Decorator** dekoriert. Der Decorator wird immer dann aufgerufen, wenn eine dekorierte Funktion ausgeführt wird. Er setzt den aktuellen State der Factory, führt die Funktionen aus und ruft den jeweils erreichten Zustand im Factory-Environment ab.

Für die MCTS werden, ähnlich wie bei der abstrakten Klasse **Production_steps**, in der abstrakten Klasse **treeNode** zwei Arten von Knoten initialisiert. Sowohl der **rootNode** (de: Start-Knoten) als auch der **ChildNode** (de: Unterknoten) erben jeweils einen Teil ihrer Attribute von der abstrakten Klasse **treeNode**. Ein **rootNode** wird von der **MCTSInIteration** mit dem Anfangszustand der Fabrik initialisiert. Ein **childNode** hingegen verwendet die **rootNode**-Instanz als Elternteil.

Die Klassen **MADQN**, **DemonstrationLogger** und **DQN_Provider** decken Funktionalitäten des MADQNs ab. Der **DemonstrationLogger** speichert State-Action-Tripel während der ersten Iteration der MCTS ohne Unterstützung des

MADQNs ab. Im nächsten Schritt werden die abgespeicherten State-Action-Tripel dazu genutzt, das **MADQN** anzulernen. Beim Training des MADQNs wird wie in Abschnitt 4.1.3 beschrieben die Total Lead Time optimiert. Da Ergebnisse von neuronalen Netzen nicht deterministisch sind und sich die Ergebnisse der verschiedenen Trainingsläufe z. B. aufgrund der Auswahl von Actions unterscheiden können, muss der bestmögliche Trainings-Checkpoint identifiziert werden. Die Auswahl des bestmöglichen Checkpoints erfolgt durch den **DQN_Provider**. Über die Schnittstelle **PUCT_Provider** wird die Wahrscheinlichkeit $P(s_t, a)$ bestimmt und an die MCTS für die Auswahl des nächsten Knotens weitergegeben.

4.2.2 MCTS-Ablauf bei der dynamischen Terminierung

In Abbildung 4.2 wird ein Überblick über die verschiedenen Schritte der MCTS in Verbindung mit dem MADQN dargestellt. Die Suche und die Evaluation neuer Knoten wird durch die **Search**-Funktion gesteuert, die so lange nach neuen Knoten sucht, bis ein vordefiniertes Zeitlimit erreicht ist. Alle neuen Knoten, die innerhalb dieses Zeitlimits entdeckt werden, können anschließend evaluiert werden. Der Knoten bzw. die jeweilige Action, die das Optimierungsziel am besten erfüllt, wird an die Factory zurückgegeben.

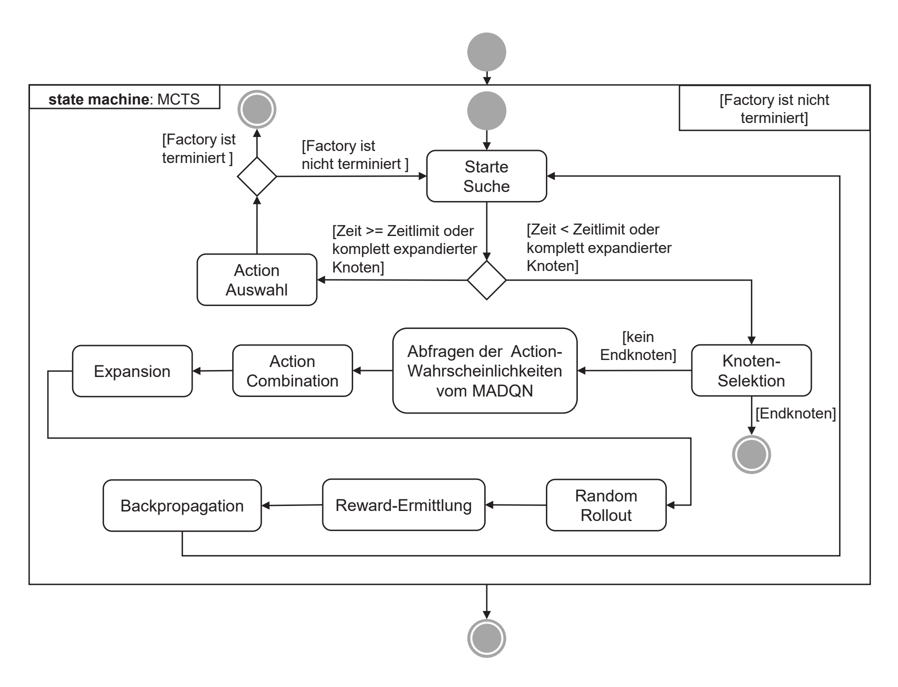


Abbildung 4.2: Ablaufdiagramm zum Prozessdurchlauf einer MCTS Iteration in Kombination mit dem MADQN

Im ersten Schritt führt, wie in Abschnitt 4.2.2 beschrieben, die Funktion **Knoten-Selektion** eine Selektion der möglichen Knoten bzw. Actions mittels der ‚Upper Confidence Bound for Trees' (UCT) durch (vgl. Abbildung 4.2). Handelt es sich bei dem ausgewählten Knoten nicht um einen End-Knoten, werden im nächsten Schritt dessen unbekannte Unterknoten untersucht. Die MCTS ohne MADQN wählt den entsprechenden Unterknoten in der Regel zufällig aus (vgl. Abschnitt 2.5.2). Mit dem MADQN wird die Action-Wahrscheinlichkeit $P(s, a)$ für eine definierte Anzahl an Unterknoten ermittelt (vgl. Abschnitt 4.2.3). Da jedem FAUF ein DQN-Agent zugewiesen wird und diese Agenten ihre Actions unabhängig voneinander treffen, ist im nächsten Schritt eine Kombination aller Actions zu einem globalen Optimum notwendig (vgl. Kapitel 4.2.4). Ist ein Unterknoten ausgewählt, erfolgt der **Rollout** (vgl. Abschnitt 2.5.2 und 4.2.2). Nach dem Rollout

folgt die **Backpropagation** (vgl. Abschnitt 2.5.2), bei der die jeweiligen Knotenwerte angepasst werden. Die neuen States, Actions und deren Rollout-Ergebnisse werden zusätzlich genutzt, um die neuronalen Netze der einzelnen DQN-Agenten anzupassen (vgl. Abschnitt 4.2.3).

Ist der Zeitrahmen für die Suche nach neuen Knoten überschritten, werden die gefundenen Knoten evaluiert. Die Knotenwerte werden mittels der UCT berechnet (vgl. Abschnitt 2.5.2). Der Knoten mit dem geringsten UCT-Wert erfüllt das in Abschnitt 4.1.3 definierte Optimierungsziel am besten und wird daher als zielführender Knoten s_{t+1} ausgewählt. Die Evaluation kann auch mittels neuronalen Netzes erfolgen. Dabei approximiert das neuronale Netz die UCT-Werte der einzelnen Knoten, basierend auf vorhergehenden UCT-Berechnungen. Die korrespondierende Action a wird dem Factory-Environment als nächste Action übergeben. Über den **FactoryWrapper** erhält die MCTS einen neuen State und startet die Funktion **Search** erneut. Dieser Ablauf wiederholt sich so oft, bis entweder alle Produkte produziert sind oder eine vordefinierte Zeit überschritten ist.

4.2.3 Funktionsweise des MADQN bei der dynamischen Terminierung

Der MCTS-Ablauf aus Abschnitt 4.2.2 wird während verschiedener Prozessschritte vom MADQN unterstützt. In diesem Abschnitt wird auf die Funktionsweise des MADQNs eingegangen. Grundlegend basiert dieser Multi-Agenten-Ansatz auf mehreren Deep-Q-Networks (DQN) (vgl. Abschnitt 2.4.5). Aufgrund der Vielzahl von FAUFs und Maschinen in einer realen Fertigung wird für einen Single-Agenten ein entsprechend großer Observation Space benötigt. Wie in Kapitel 2.4.4 beschrieben, ist ein Single-Agent aber nicht in der Lage, in komplexeren Szenarien zu lernen, und ist zusätzlich nicht skalierbar. Aus diesem Grund wird das DQN in dieser Methode zu einem Multi-Agenten-System erweitert, bei dem jeder FAUF seinen eigenen Agenten besitzt.

Die für das Anlernen der verschiedenen neuronalen Netze des MADQNs benötigten Trainingsdaten werden teilweise von der MCTS erzeugt. Dafür erstellt die MCTS zunächst einen Entscheidungsbaum ohne Unterstützung des MADQNs. Da das MADQN ein Off-policy-Algorithmus ist, besitzt es einen Replay-Buffer, der während der Exploration-Phase der MCTS mit Trainingsdaten befüllt wird (vgl. Abschnitt 2.4.5). Nach der Exploration-Phase werden die entstandenen Trainingsdaten für das Lernen des MADQNs genutzt. Den Replay-Buffer füllt die MCTS mit ihren erzeugten Trainingsdaten und unterstützt so das MADQN beim Training, da sie mit diesen Trainingsdaten den möglichen Lösungsraum verkleinert. Um ein Overfitting auf die MCTS-Trainingsdaten zu vermeiden, befüllt das MADQN den Replay-Buffer teilweise auch mit selbst-erzeugten Trainingsdaten. Das Verhältnis zwischen MCTS- und MADQN-Trainingsdaten muss vor dem Training festgelegt werden. Es wird angenommen, dass das MAQDN mit den Trainingsdaten der MCTS nicht nur schneller lernt, sondern auch ein besseres Lernergebnis erzielt. Zusätzlich kann der Replay-Buffer des MADQNs auch mit dem Terminierungsergebnis eines APS-Systems befüllt werden, das einen nahezu optimalen Ablauf einer ungestörten Fertigung darstellt.

Ist der DQN-Agent eines FAUFs angelernt, kann dieser auf Basis des States s_t und der Action a die Wahrscheinlichkeit $P(s_t, a)$ approximieren, mit der diese Action in dem gewählten State einen optimalen Fertigungsdurchlauf für diesen FAUF darstellt. Eine Action entspricht in diesem Fall der Auswahl eines FAUF-Agenten, den nächsten AVO an einer bestimmten Maschine durchführen zu lassen. Auf diese Art und Weise kann das FAUF-DQN die Wahrscheinlichkeit, ob die Action bzw. Maschine eine gute Wahl im Sinne des Optimierungszieles ist, für jede Maschine approximieren. Die einzelnen FAUF-DQNs ermitteln die Action-Wahrscheinlichkeiten unabhängig voneinander. Ausgegeben werden die Action-Wahrscheinlichkeiten aller FAUFs, die die einzelnen FAUF-DQNs approximiert haben, in einer Matrix (vgl. Abbildung 4.3). Jeder Zeile in der Ausgabematrix entspricht dabei der Ausgabe des spezifischen FAUF-DQNs. In diesem Beispiel besteht das Factory-Environment aus sechs Maschinen ($M_1 - M_6$) und 10 FAUFs.

M_1 M_2 M_3 M_4 M_5 M_6

FAUF-DQN

[[0.22, 0.73, 0.05, 0.91, 0.27, 0.22], $FAUF_0$
[0.80, 0.55, 0.84, 0.73, 0.37, 0.70], $FAUF_1$
[0.22, 0.25, 0.74, 0.11, 0.98, 0.18], $FAUF_2$
[0.54, 0.51, 0.11, 0.14, 0.76, 0.77], $FAUF_3$
[0.70, 0.86, 0.55, 0.52, 0.04, 0.81], $FAUF_4$
[0.82, 0.97, 0.55, 0.98, 0.41, 0.08], $FAUF_5$
[0.87, 0.83, 0.39, 0.11, 0.52, 0.16], $FAUF_6$
[0.50, 0.01, 0.75, 0.86, 0.83, 0.11], $FAUF_7$
[0.09, 0.86, 0.98, 0.62, 0.67, 0.22], $FAUF_8$
[0.86, 0.66, 0.96, 0.54, 0.75, 0.80]] $FAUF_9$

Action-Wahrscheinlichkeit

Abbildung 4.3: Ausgabematrix der Action-Wahrscheinlichkeiten vom MADQN für 10 FAUFs und 6 Maschinen

Abhängig von der Anzahl der FAUFs und den vorhandenen Hardwarebeschränkungen beim Training kann es sinnvoll sein, mehrere ähnliche FAUFs in einem DQN-Agenten zusammenzufassen. Damit kann die Anzahl der Agenten beim Training minimiert werden, was zu einer geringeren Hardwareauslastung führt. Die Ähnlichkeit der FAUFs lässt sich auf Basis der Arbeitspläne bestimmen. FAUFs, die die selben Maschinen für ihre Abarbeitung benötigen, werden dabei einem DQN-Agenten zugeordnet. Aufgrund der großen Ähnlichkeit dieser FAUFs bzw. ihrer Arbeitspläne kann der DQN-Agent eine Policy lernen, die die Total Lead Time aller zusammengefassten FAUFs minimiert.

Ist das MADQN angelernt, kann es in Kombination mit der MCTS für die dynamische Terminierung genutzt werden. In Abbildung 4.4 ist das Zusammenwirken von MADQN und MCTS schematisch abgebildet.

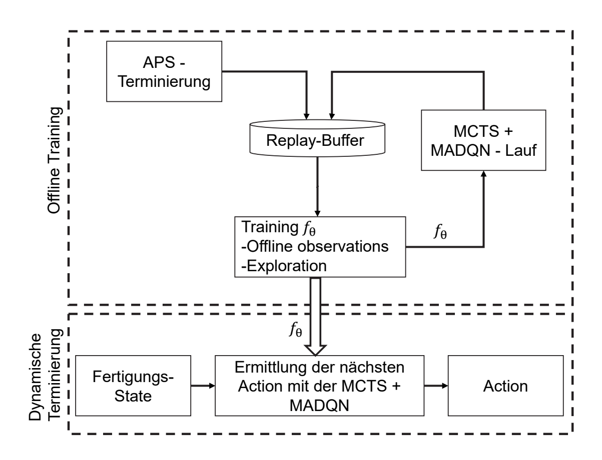


Abbildung 4.4: Zusammenwirken Offline Training und dynamische Terminierung

Der Replay-Buffer (vgl. Abbildung 4.4) für das Training des MADQNs wird mittels der vorhandenen APS-Terminierung und selbst erzeugten Trainingsdaten der MCTS befüllt. Innerhalb mehrerer Iterationen wird das MADQN angelernt. Die Action-Wahrscheinlichkeiten des MADQNs werden mit jenen der MCTS verglichen und bewertet. In vorgeschriebenen Abständen werden die Netzparameter des MADQNs angepasst; d. h., das MADQN lernt, die Action-Wahrscheinlichkeiten besser zu approximieren (vgl. Abschnitt 2.4.5). Ist das MADQN angelernt, kann es für die dynamische Terminierung eingesetzt werden. Dafür dient als Ausgangspunkt ein State aus dem Factory-Environment. MADQN und MCTS ermitteln wie in Abschnitt 4.2.2 beschrieben die beste Action bzw. die beste Maschine für den nächsten Fertigungsschritt. Dabei werden die Action-Wahrscheinlichkeiten mit dem MADQN wie in Abbildung 4.3 dargestellt bestimmt.

4.2.4 Lösung des Action-Combination-Problems des MADQNs

Jeder Agent des MADQNs ist einem festen FAUF zugeordnet (vgl. Abschnitt 4.2.3). Wird dem DQN-Agenten, nachfolgend repräsentiert durch die Netzparameter f_θ, ein State s_i für einen FAUF übergeben, gibt er einen Vektor mit den Wahrscheinlichkeiten p aller Actions zurück (vgl. Abbildung 4.3 und Formel 4.1).

$$f_\theta(s_i) = [p(s_i, a_1, \theta), \ldots, p(s_i, a_{K+1}, \theta)]^T \tag{4.1}$$

Die Actions umfassen dabei alle Maschinen $\mathcal{K} = \{1, \ldots, K\}$ und zusätzlich eine Action, um keine Maschine auszuwählen. Daraus ergeben sich $K + 1$ mögliche Actions. Wie in Abschnitt 2.4.4 beschrieben, werden die Action-Wahrscheinlichkeiten p von den einzelnen FAUF-Agenten unabhängig voneinander getroffen - das bedeutet, dass ein FAUF-Agent bei der Ermittlung der Action-Wahrscheinlichkeiten nur seine eigenen Actions betrachtet und damit seinen eigenen Reward maximiert. Das hat zur Folge, dass z. B. für einen Zeitschritt mehrere FAUF-Agenten unabhängig voneinander die gleiche Action und damit die gleiche Maschine auswählen. Andere Maschinen, die diese FAUF-Agenten auch hätten aussuchen können, werden in diesem Zeitschritt gar nicht ausgewählt. Ein globales Optimum ist mit diesem Vorgehen nicht erreichbar werden. Wie in Abschnitt 3.2.4 beschrieben, muss dieses Problem beim Einsatz von Multi-Agenten-Methoden gelöst werden. Veröffentlichungen, die das Action-Combination-Problem beschreiben und lösen, sind nicht bekannt. Nachfolgend wird das Action-Combination-Problem mathematisch definiert und so umgeformt, dass es mit einem Linear Integer Program gelöst werden kann.

Für die einzelnen Action-Wahrscheinlichkeiten der FAUF-Agenten kann angenommen werden, dass diese immer größer als 0 sind: $p(s_i, a_j, \theta) > 0 \; \forall j \in \{1, \ldots, K + 1\}$. Die Action mit der höchsten Action-Wahrscheinlichkeit des

i-ten FAUF-Agenten wird definiert als: $a^{(i)} \in \{a_1, \ldots, a_{K+1}\}$. Die Kombination aller höchsten Action-Wahrscheinlichkeiten aller FAUF-Agenten ist $a = \left(a^{(1)}, \ldots, a^{(N)}\right)$. Die einzelnen Action-Wahrscheinlichkeiten der FAUF-Agenten sind dabei unabhängig voneinander, weshalb für die Gesamt-Wahrscheinlichkeit aller kombinierten Action-Wahrscheinlichkeiten $p(s, a, \theta)$ Gleichung 4.2 gilt.

$$p(s, a, \theta) \approx \prod_{i=1}^{N} p(s_i, a^{(i)}, \theta) \tag{4.2}$$

Die Gesamt-Wahrscheinlichkeit wird von der MCTS während des Selektions-Schritts genutzt (vgl. Abschnitt 2.5.4 und Formel 2.9). Da nicht alle Actions des States s sinnvoll sind, z. B. wenn sich ein FAUF-Agent für eine Maschine entscheidet, auf der er nicht bearbeitet werden darf, wird ein Branching-Factor (de: Verzweigungsfaktor) $|\mathcal{A}|$ eingeführt. Dabei ist $\mathcal{A}(s)$ eine Teilmenge aller möglichen Actions a in State s. Alle Actions werden dafür anhand ihrer Action-Wahrscheinlichkeiten $p(s, a, \theta)$ bewertet. Actions mit einer hohen Action-Wahrscheinlichkeit werden hierbei als diejenigen Actions definiert, die aus FAUF-Sicht am sinnvollsten sind. Je größer der Branching-Factor ist, desto mehr Actions bzw. Action-Kombinationen werden untersucht.

Das Action-Combination-Problem ist ein kombinatorisches Optimierungsproblem, bei dem die beste Action-Kombination aller Actions ermittelt werden muss. Nachfolgend wird das Action-Combination-Problem mathematisch beschrieben und anschließend gelöst. Dafür wird der Action-Vektor x_i des i-ten FAUF-Agenten als Vektor mit $K + 1$ Einträgen definiert, bei dem alle Einträge den Wert 0 annehmen, außer der Eintrag der gewählten Action $a^{(i)} = a_j$. Die gewählte Action ist somit der Eintrag $x_{i,j} = 1 \Leftrightarrow a^{(i)} = a_j$ aus $x_i \in \{0, 1\}^{K+1}$. Damit kann Folgendes für Maschinen mit Warteschlangen definiert werden (vgl. Formel 4.3a-4.3b):

$$\underset{x_i \in \{0,1\}^{K+1}}{\arg\max} \prod_{i=1}^{N} x_i^T f_\theta(s_i) \tag{4.3a}$$

$$\text{erfüllt} \quad \sum_{j=1}^{K+1} x_{i,j} = 1 \quad \forall i \in \mathcal{P} \tag{4.3b}$$

Das in Gleichung 4.3a und 4.3b formulierte Problem ist ein nicht lineares, konvexes Problem. Um das vorliegende Optimierungsproblem mit effizienten Optimierungsverfahren zu lösen, muss dieses als ein lineares, konvexes Optimierungsproblem, ein sogenanntes Linear Integer Program, umgeformt werden. Aus diesem Grund wird die Zielfunktion aus Gleichung 4.3a in Gleichung 4.4 umformuliert.

$$\underset{x_i \in \{0,1\}^{K+1}}{\arg\max} \prod_{i=1}^{N} x_i^T f_\theta(s_i) \equiv \underset{x_i \in \{0,1\}^{K+1}}{\arg\max} \sum_{i=1}^{N} \log\left(x_i^T f_\theta(s_i)\right) \tag{4.4}$$

Gleichung 4.4 erfüllt folgende Bedingungen: $x_i \in \{0,1\}^{K+1}$ und $[f_\theta(s_i)]_j > 0$. Beide Bedingungen stellen sicher, dass die Logarithmusfunktion in Gleichung 4.4 niemals 0 wird. Folgendes kann daher angenommen werden:

$$\begin{aligned} \log\left(x_i^T f_\theta(s_i)\right) &= \log \sum_{j=1}^{K+1} x_{i,j} [f_\theta(s_i)]_j \\ &= \sum_{j=1}^{K+1} x_{i,j} \log\left[f_\theta(s_i)\right]_j \end{aligned} \tag{4.5}$$

Da ein FAUF-Agent nur eine Action wählen kann und damit immer nur ein Eintrag in x_i den Wert 1 annimmt, kann der Logarithmus auf alle Elemente von $f_\theta(s_i)$ aufgeteilt werden. Daraus ergibt sich: $\hat{f}_\theta(s_i) = [\hat{p}(s_i, a_1, \theta), \ldots, \hat{p}(s_i, a_{K+1}, \theta)]^T$, wobei Folgendes angenommen wird:

$$\hat{p}(s_i, a_j, \theta) = \log p(s_i, a_j, \theta) \tag{4.6}$$

Das Optimierungsproblem kann somit wie folgt aufgestellt werden (vgl. Gleichung 4.7a und 4.7b).

$$\underset{x_i \in \{0,1\}^{K+1}}{\arg\max} \sum_{i=1}^{N} x_i^T \hat{f}_\theta(s_i) \tag{4.7a}$$

$$\text{erfüllt} \quad \sum_{j=1}^{K+1} x_{i,j} = 1 \quad \forall i \in \mathcal{P} \tag{4.7b}$$

Das vorliegende Optimierungsproblem ist jetzt linear und konvex und kann daher mit einem Linear Integer Program gelöst werden. Im nächsten Schritt wird das Optimierungsproblem gelöst. Dafür wird ein iterativer Ansatz genutzt, um für die Actions in $|\mathcal{A}|$ die beste Kombination zu finden. Das Ergebnis der l-ten Action-Kombination ist $\hat{x}_{i,l}$. Damit kann das finale Linear Integer Program wie in den Gleichungen 4.8a- 4.8c formuliert werden.

$$\hat{x}_{i,l} = \underset{x_i \in \{0,1\}^{K+1}}{\arg\max} \sum_{i=1}^{N} x_i^T \hat{f}_\theta(s_i) \tag{4.8a}$$

$$\text{erfüllt} \quad \sum_{j=1}^{K+1} x_{i,j} = 1 \quad \forall i \in \mathcal{P}, \tag{4.8b}$$

$$\sum_{i=1}^{N} x_i^T \hat{x}_{i,m} \leq N - 1 \quad \forall m \in \{1, \ldots, l\} \setminus \{l\} \tag{4.8c}$$

Für Maschinen ohne Warteschlange muss noch eine Bedingung hinzugefügt werden, die es nicht erlaubt, dass mehrere FAUFs dieselbe Maschine auswählen (vgl. Gleichung 4.9).

$$\sum_{i=1}^{N} x_{i,j} = 1, \quad \forall j \in \mathcal{K} \tag{4.9}$$

4.2.5 Clustering der FAUFs für die dynamische Terminierung

Da unterschiedliche Fertigungen in der Regel in der Anzahl der Maschinen und Bauteile voneinander abweichen, muss die in dieser Arbeit entwickelte Methode für die dynamische Terminierung skalierbar sein. Die Größe des Lösungsraums nimmt mit steigender Anzahl an Maschinen und/oder Bauteilen exponentiell zu. Obwohl die MCTS in Kombination mit dem MADQN in einem Lösungsraum besser gute Actions findet als ohne, ist davon auszugehen, dass die Lösungsqualität bei wachsender Lösungsraumgröße und gleichbleibender Berechnungszeit abnimmt. In diesem Abschnitt wird daher darauf eingegangen, wie die verschiedenen FAUFs einzelnen Clustern zugewiesen werden können. Anschließend ist es

möglich, diese Cluster parallel nach den besten Actions zu durchsuchen. Theoretisch lassen sich so beliebig viele Cluster kombinieren, sodass gute Actions auch in sehr großen Fertigungen gefunden werden können.

4.2.5.1 Ablauf des Clusterings der FAUFs

Das Clustering von FAUFs lässt sich in verschiedene Schritte unterteilen, die nachfolgend erklärt werden. Ein Überblick über den Ablauf des Clusterings wird in Abbildung 4.5 aufgezeigt.

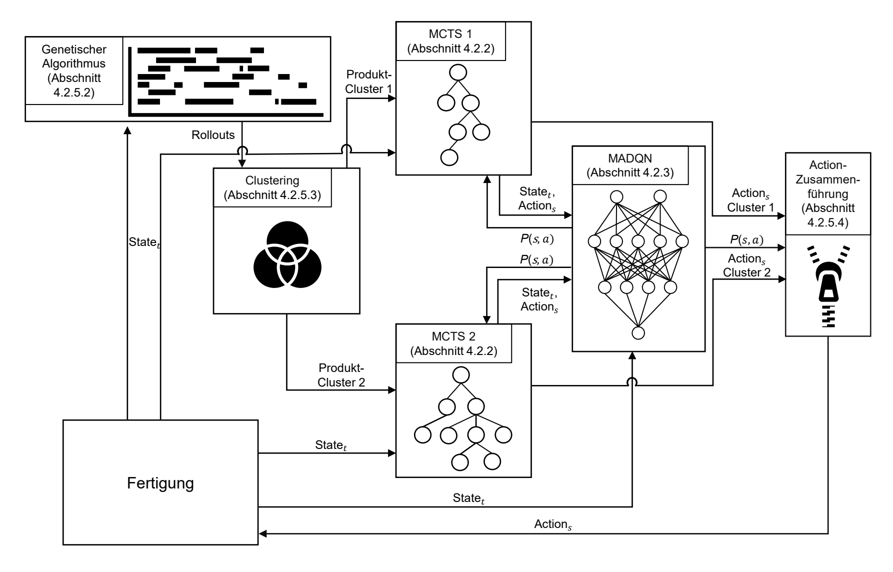


Abbildung 4.5: Überblick über die einzelnen Schritte des Clusterns der FAUFs im Rahmen der dynamischen Terminierung

Zuerst wird dafür, wie in Abbildung 4.5 dargestellt, ein State des Factory-Environments an einen genetischen Algorithmus (vgl. Abschnitt 3.2.3) übergeben. Ziel ist es eine festgelegte Anzahl von bestmöglichen Fertigungssequenzen zu erzeugen. Unter einer Fertigungssequenz wird in diesem Zusammenhang ein kompletter Fertigungsablauf verstanden; d. h., alle FAUFs werden vom aktuellen

Fertigungsfortschritt ausgehend weiterbearbeitet, bis sie fertig bearbeitet sind (vgl. Abbildung 4.6). Dabei durchlaufen die FAUFs die Maschinen, die den jeweiligen Arbeitsvorgängen im Arbeitsplan zugeordnet sind. Sind alle FAUFs gefertigt, ist eine Fertigungssequenz abgeschlossen.

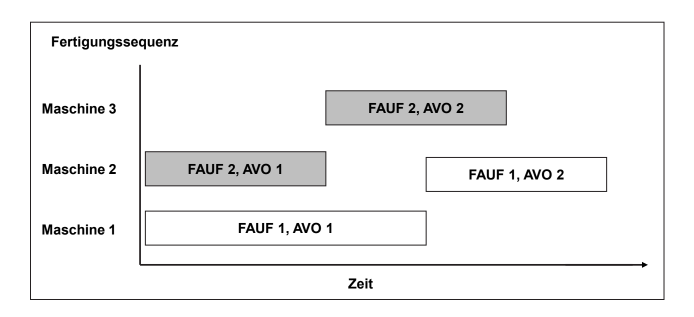


Abbildung 4.6: Sequenzierung der einzelnen FAUF-AVOs zu einer Fertigungssequenz

Die Fertigungssequenzen eines FAUFs unterscheiden sich aufgrund der gewählten Maschinen und der Fertigungszeiten. So entsteht eine zweite Fertigungssequenz, wenn ein FAUF z. B. an einer anderen Maschine als in der ersten Fertigungssequenz gefertigt wird, oder sich die Reihenfolge der Warteschlangen ändert. Im OSSP können daher aufgrund der vielen Freiheitsgrade extrem viele verschiedene Fertigungssequenzen gebildet werden. Von dieser Menge an Fertigungssequenzen sind aber nur sehr wenige sinnvoll und führen zu einem Fertigungsergebnis, das das festgelegte Optimierungsziel erfüllt (vgl. Abschnitt 4.1.3). Ein Clustern aller Fertigungssequenzen würde ein schlechtes Lösungsergebnis hervorbringen, da die ‚schlechten' Fertigungssequenzen zahlenmäßig viel stärker vertreten sind als die zielführenden. Mit Hilfe des genetischen Algorithmus wird eine festgelegte Anzahl von Fertigungssequenzen bestimmt, die mit einer hohen Wahrscheinlichkeit das Fertigungsergebnis im Sinne des Optimierungsziels optimieren (vgl. Abschnitt 4.2.5.2).

Die so ermittelten Fertigungssequenzen können anschließend für das Clustern (vgl. Abschnitt 4.2.5.3) genutzt werden. Dabei werden Cluster von FAUFs gebildet, die über eine möglichst hohe Schnittmenge bzw. Ähnlichkeit verfügen (vgl.

Abschnitt 2.6.2). FAUFs, die zur gleichen Zeit die gleichen Maschinen benötigen, werden somit in einem Cluster vereint. Die Annahme der Existenz von FAUFs mit kleinen und großen Schnittmengen lässt sich dadurch begründen, dass selbst in einer Fertigung, die nur eine Art von Bauteil fertigt, die verschiedenen FAUFs unterschiedliche Fertigungszustände aufweisen. Das heißt, dass bei einigen FAUFs die Fertigung gerade erst startet und diese FAUFs somit noch viele Arbeitsvorgänge durchlaufen müssen, die sie eventuell auch noch an verschiedenen alternativen Maschinen abarbeiten können. Andere FAUFs sind am Ende der Fertigung und haben nur noch ein bis zwei Arbeitsvorgänge bis zur Fertigstellung. Die Schnittmenge der FAUFs mit unterschiedlichem Fertigungsfortschritt ist dementsprechend gering und so können mindestens zwei unabhängige Cluster gebildet werden.

Anschließend wird für jedes Cluster eine MCTS erzeugt, die von einem MADQN unterstützt wird - analog zu dem im Abschnitt 4.2.1 vorgestellten Ablauf. Obwohl die Ähnlichkeit der FAUFs clusterübergreifend möglichst gering ist, kann es dennoch vorkommen, dass zwei unterschiedliche FAUFs aus den beiden Clustern im nächsten Schritt an dieselbe Maschine geschickt werden, weswegen eine Action-Zusammenführung notwendig ist (vgl. Abschnitt 4.2.5.4).

4.2.5.2 FAUF-Sequenzierung mit Hilfe eines genetischen Algorithmus

Die Fertigungssequenzerstellung liefert die Grundlage für das nachfolgende Clustern der FAUFs. Die FAUF-AVOs werden bei der Fertigungssequenzerstellung den frühestmöglichen Start-Zeiten an den Maschinen zugeordnet (vgl. Abbildung 4.6). Da nachfolgend aber nicht die Maschinen, sondern die FAUFs geclustert werden sollen, müssen die Fertigungssequenzen in FAUF-Sequenzen umgewandelt werden (vgl. Abbildung 4.7).

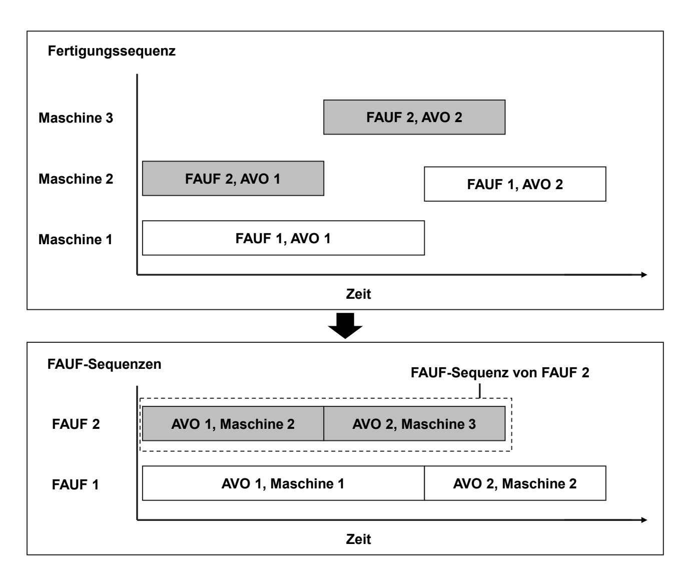


Abbildung 4.7: Umwandlung einer Fertigungssequenz in FAUF-Sequenzen

Da die Anzahl an möglichen Fertigungssequenzen und damit auch FAUF-Sequenzen bei einem OSSP extrem groß wird und der überwiegende Teil der FAUF-Sequenzen nicht dem Optimierungsziel der Total Lead Time-Optimierung folgt, müssen zunächst die bestmöglichen FAUF-Sequenzen der Bauteile ermittelt werden. Die Qualität der ausgewählten FAUF-Sequenzen entscheidet maßgeblich über die Qualität des Clusterns und hat somit einen großen Einfluss auf die anschließende Lösungsqualität. In einer Fertigung mit sehr wenigen Maschinen und Freiheitsgraden ist auch die Anzahl der möglichen Fertigungssequenzen entsprechend gering, sodass es Fälle gibt, bei denen die besten FAUF-Sequenzen mit einfachen Heuristiken gefunden werden können. Für Fertigungen mit extrem vielen Freiheitsgraden, bei denen die Anzahl der möglichen Fertigungssequenzen und damit FAUF-Sequenzen extrem groß ist, kann ein genetischer Algorithmus gute Fertigungssequenzen bzw. FAUF-Sequenzen ermitteln. Die Ablaufschritte des genetischen Algorithmus für die FAUF-Sequenzerstellung sind in Abbildung 4.8 dargestellt.

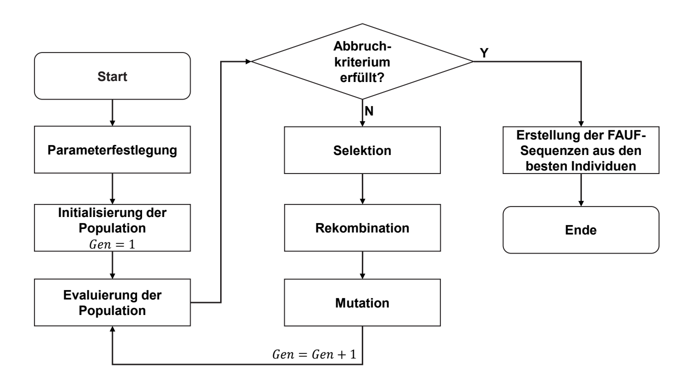


Abbildung 4.8: Ablauf der FAUF-Sequenzerstellung mittels genetischen Algorithmus

Im ersten Schritt werden zentrale Parameter für den genetischen Algorithmus festgelegt (vgl. Abschnitt 3.2.3). Zunächst wird die Anzahl von Individuen in einer Population definiert. Individuen entsprechen in der vorliegenden Problemstellung den unterschiedlichen FAUF-Sequenzen. Zusätzlich wird ein Abbruchkriterium bestimmt. Im Anschluss daran kann die Population der *1.Generation* initialisiert werden (vgl. Abbildung 4.8). Die Initialisierung erfolgt dabei zufällig und erzeugt eine festgelegte Anzahl an FAUF-Sequenzen. Die FAUF-Sequenzen werden dann anhand des Optimierungsziels, also der Total Lead Time-Optimierung, verglichen und eingeordnet. Das Abbruchkriterium ist eine vorher definierte Anzahl an Generationen, die durchlaufen werden müssen. Ist das Abbruchkriterium nicht erfüllt, wird eine neue Generation an FAUF-Sequenzen erzeugt. Dafür werden die besten FAUF-Sequenzen ausgewählt und in die nächste Generation übernommen. Zusätzlich werden diese FAUF-Sequenzen rekombiniert und mutiert und bilden so die nächste Generation. Anschließend werden die auf diese Weise neu gebildeten FAUF-Sequenzen evaluiert und es wird geprüft, ob das Abbruchkriterium erfüllt ist. Ist das der Fall, werden die so erzeugten FAUF-Sequenzen der letzten Generation als Grundlage für das nachfolgende Clustern genutzt. Die Ausgabe der Rollouts erfolgt anhand der in Abbildung 4.9 dargestellten Daten-Repräsentation.

Maschinen-ID

FAUF-Sequenz für FAUF 0

[[0, 3, 3, 3, 3, nan, nan, nan, nan, 5, 5, 5, 6, 6, …, 4],
[0, nan, nan, nan, nan, nan, nan, nan, nan, 3, 3, 3, 3, 3, …, 6],
⋮
[9, 6, 6, 3, 3, 3, 3, nan, 2, 6, nan, nan, nan, nan, …, nan]]

i_0 i_1 i_2 i_3 i_4 i_5 i_6 i_7 i_8 i_9 i_{10} i_{11} i_{12} i_{13} i_T

FAUF-ID

Zeitschritte in der Fertigung

Abbildung 4.9: Daten-Repräsentation der FAUF-Sequenzen

Jede Zeile des Arrays in Abbildung 4.9 repräsentiert dabei eine FAUF-Sequenz. In der ersten Spalte werden die FAUF-IDs angegeben. Aufgrund der vielen Fertigungsablaufmöglichkeiten ergeben sich für jeden FAUF mehrere FAUF-Sequenzen, weshalb in Abbildung 4.9 die ersten beiden FAUF-Sequenzen dem FAUF mit der ID 0 zugeordnet sind. Die einzelnen Einträge in einer FAUF-Sequenz beschreiben entweder die Maschinen-ID an der der FAUF zum Zeitpunkt i bearbeitet wird, oder beinhalten einen ‚nan'-Eintrag (en: not a number), wenn der FAUF transportiert wird oder in einer Warteschlange steht und somit nicht gefertigt werden kann.

4.2.5.3 Clustern der Rollouts

Die Datengrundlage für das Clustern der FAUFs liefert der im Abschnitt 4.2.5.2 beschriebene genetische Algorithmus. Nachfolgend werden die FAUF-Sequenzen geclustert (vgl. Abschnitt 2.6). Dafür müssen die FAUF-Sequenzen zunächst auf ihre zeitliche und räumliche Schnittmenge untersucht werden. Die Ermittlung der Schnittmenge erfolgt mittels einer Distanz-Metrik (vgl. Abschnitt 2.6.3). Anschließend werden die FAUF-Sequenzen anhand ihrer Schnittmengen Clustern zugeordnet. Als letzten Schritt beim Clustern sind nun die FAUFs den Clustern zuzuweisen.

Ähnlichkeits-Metrik

Um die FAUFs einzelnen Clustern zuordnen zu können, müssen zunächst die einzelnen FAUF-Sequenzen auf Schnittmengen bzw. Ähnlichkeiten untersucht werden (vgl. Abschnitt 2.6.2). Eine Schnittmenge zwischen mindestens zwei FAUF-Sequenzen tritt immer dann auf, wenn mehrere unterschiedliche FAUFs zur selben Zeit an derselben Maschine gefertigt werden sollen. In diesem Fall ist auch von einem Konflikt zu sprechen. In Abbildung 4.10 werden zwei FAUF-Sequenzen auf Ähnlichkeiten bzw. Konflikte untersucht. Dabei werden zwei Konflikte identifiziert, da die beiden FAUFs (mit der ID 0 und ID 1) in den entsprechenden FAUF-Sequenzen zur selben Zeit sowohl Maschine 3 als auch Maschine 5 belegen.

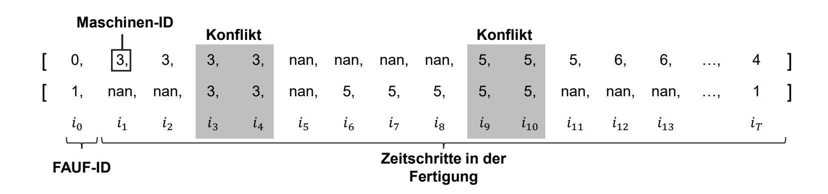


Abbildung 4.10: Konfliktidentifikation zwischen den FAUF-Sequenzen

Um die Ähnlichkeit der FAUF-Sequenzen festzustellen, wird zunächst die Jaccard-Metrik bestimmt, die die Ähnlichkeit zweier Mengen, im vorliegenden Fall FAUF-Sequenzen, erfasst (vgl. Abschnitt 2.6.2). Diese Metrik gibt den sogenannten Abstand zweier FAUF-Sequenzen an. Die einzelnen Einträge der FAUF-Sequenzen müssen für die Berechnung der Jaccard-Metrik zunächst umgeformt werden. Ohne diese Umformung der FAUF-Sequenzen würden bei der Ermittlung der Jaccard-Metrik zeitunabhängige Schnittmengen zwischen den FAUF-Sequenzen gefunden werden, wenn z. B. zwei FAUFs innerhalb der FAUF-Sequenzen dieselbe Maschine nutzen - unabhängig davon, ob es eine zeitliche Überlappung gibt. Aus diesem Grund müssen die Werte der FAUF-Sequenzen-Einträge auch eine zeitliche Komponente beinhalten. Formel 4.10 beschreibt die Berechnung des neuen FAUF-Sequenz-Werts V_t^{neu} zum Zeitschritt i_t.

$$V_t^{neu} = i_T \cdot V_t + i_t \tag{4.10}$$

Der nicht angepasste FAUF-Sequenz-Eintrag zum Zeitpunkt i_t wird mit V_t angegeben. Die komplette Makespan der FAUF-Sequenz wird als i_T definiert. Mit Hilfe der Formel 4.10 werden neue FAUF-Sequenz-Einträge für alle Einträge berechnet, die kein ‚nan' beinhalten. Mit der Jaccard-Metrik kann nachfolgend die Ähnlichkeit der einzelnen FAUF-Sequenzen in einer Matrix ausgegeben werden (vgl. Abbildung 4.11).

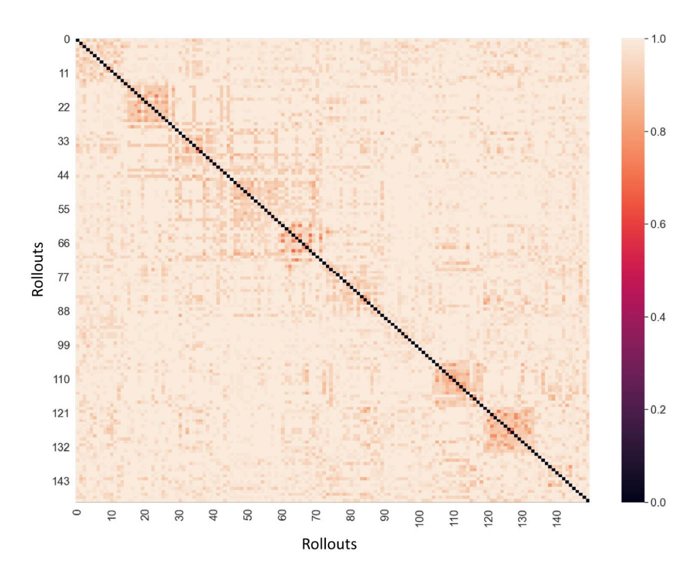


Abbildung 4.11: Jaccard-Distanz zwischen den einzelnen FAUF-Sequenzen

Auf der X- und Y-Achse in Abbildung 4.11 sind jeweils alle FAUF-Sequenzen (in diesem Beispiel 150 Stück) abgebildet. Die Distanz zweier FAUF-Sequenzen zueinander kann über die Farbe der entsprechenden Koordinateneinträge abgelesen werden. Je größer die Schnittmenge zwischen den FAUF-Sequenzen ist, desto dunkler wird das entsprechende Feld in der Matrix dargestellt. Im nächsten Schritt werden die so ermittelten Ähnlichkeiten der einzelnen FAUF-Sequenzen für die Clustererstellung genutzt.

Clustering-Algorithmus

Nach der Berechnung der Ähnlichkeit bzw. der Distanz der einzelnen FAUF-Sequenzen zueinander findet im nächsten Schritt das Clustering der FAUF-Sequenzen statt (vgl. Abschnitt 2.6.3). Das hierarchisch-agglomerative Clusterverfahren ermöglicht das Clustern der FAUF-Sequenzen, ohne wie bei anderen Clusterverfahren, einen Startpunkt für das Clustern zu definieren oder die Clusteranzahl bereits vor dem Clustern festlegen zu müssen. Jede FAUF-Sequenz wird zu Beginn des hierarchisch-agglomerativen Clusterverfahrens als einzelnes Cluster betrachtet. Zwischen den einzelnen Clustern wird dann die Distanz berechnet und Cluster, die eine geringe Distanz zueinander aufweisen, werden zu einem Cluster zusammengeführt. Dieser Vorgang wird so lange wiederholt, bis ein Cluster mit allen FAUF-Sequenzen erreicht ist. Das Ergebnis dieses Clusterverfahrens kann in einem Dendrogramm abgebildet werden (vgl. Abbildung 4.12).

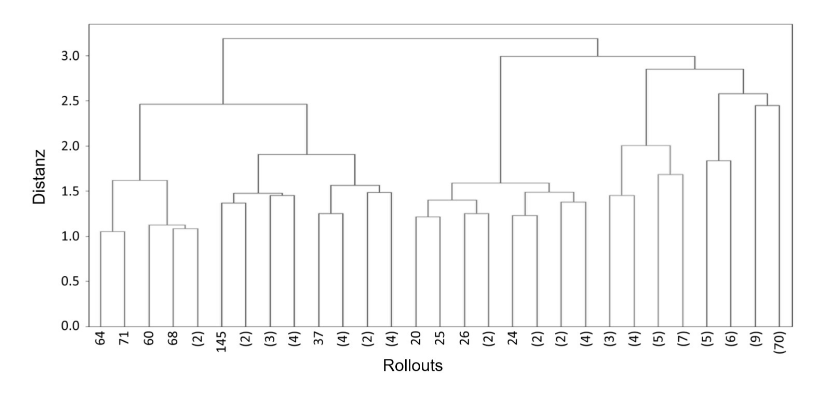


Abbildung 4.12: Hierarchisch-agglomeratives Clustern der FAUF-Sequenzen

Auf der X-Achse in Abbildung 4.12 sind die entsprechenden FAUF-Sequenzen angeordnet. Zur besseren Veranschaulichung sind die untersten Ebenen des Dendrogramms nicht dargestellt. Das führt dazu, dass in diesem Schaubild die FAUF-Sequenzen bereits teilweise zusammengefasst sind. Einzelne FAUF-Sequenzen werden auf der X-Achse mit ihrer FAUF-Sequenz-Nummer gekennzeichnet. Wurden FAUF-Sequenzen bereits zu Clustern zugeordnet, gibt eine Zahl in Klammern die Anzahl der FAUF-Sequenzen in diesem Cluster an. Die Y-Achse in Abbildung 4.12 gibt die Distanzgrenze an, ab der FAUF-Sequenzen zusammengefasst werden. Je größer die Distanzgrenze ist, desto mehr FAUF-Sequenzen werden zusammengefasst.

Im nächsten Schritt wird die optimale Anzahl an Clustern mittels des Silhouetten Indexes bestimmt (vgl. Abschnitt 2.6.4). Dafür wird für die verschiedenen Anzahlen an Clustern der jeweilige Silhouetten Index berechnet. Die Cluster-Anzahl mit dem höchsten Silhouetten Index besitzt die beste Aufteilung. In den nachfolgenden Schritten werden die so ermittelten Cluster für die Feststellung der besten Actions für die verschiedenen FAUFs genutzt.

FAUF-Zuordnung zu entsprechendem Cluster

Nachdem die verschiedenen FAUF-Sequenzen einzelnen Clustern zugeordnet sind, müssen nachfolgend die FAUFs den Clustern zugewiesen werden. Da für jeden FAUF mehrere FAUF-Sequenzen ermittelt wurden, sind die unterschiedlichen FAUF-Sequenzen eines FAUFs mehreren Clustern zuordenbar - das Bauteil bzw. der FAUF selbst kann aber letztlich nur einem Cluster zugeteilt werden. Um nun das entsprechende Bauteil einem Cluster zuzuordnen, wird die Anzahl der FAUF-Sequenzen eines FAUFs in jedem Cluster aufsummiert und mit den anderen Clustern verglichen (vgl. Abbildung 4.13).

	$FAUF_0$	$FAUF_1$	$FAUF_2$	$FAUF_3$	$FAUF_4$	$FAUF_5$	$FAUF_6$	$FAUF_7$	$FAUF_8$	$FAUF_9$
Cluster 1:	10	11	12	12	9	3	5	0	0	1
Cluster 2:	3	1	1	0	3	12	10	14	15	12
Cluster 3:	2	3	2	3	3	2	0	1	0	2

Abbildung 4.13: Zuordnung der FAUF-Sequenzen zu den entsprechenden Clustern

So werden zehn FAUF-Sequenzen für $FAUF_0$ dem Cluster 1, drei dem Cluster 2 und zwei dem Cluster 3 zugeordnet. Da die meisten Rollouts für Bauteil $FAUF_0$ dem Cluster 1 zugerechnet werden, wird das Bauteil bzw. der FAUF dem Cluster 1 zugeteilt - analog dazu werden auch die anderen Bauteile den Clustern zugeordnet. Daraus ergeben sich für die zehn Bauteile zwei Cluster, wobei $FAUF_0 - FAUF_3$ Cluster 1 und $FAUF_4 - FAUF_9$ Cluster 2 zugewiesen werden. Dem dritten Cluster wird in diesem Beispiel kein Bauteil zugeordnet. Anschließend wird für jedes Cluster ein Array erstellt und damit zuletzt jeweils eine MCTS initialisiert.

4.2.5.4 Action-Zusammenführung

Nach der Zuordnung der FAUFs zu den einzelnen Clustern müssen im nächsten Schritt die gewählten Actions der einzelnen Cluster-MCTS zusammengeführt werden. Da die Cluster-MCTS bei der Auswahl der besten Actions nur die im Cluster befindlichen FAUFs betrachtet, kann es vorkommen, dass FAUFs von

unterschiedlichen Clustern gleichzeitig an dieselbe Maschine geschickt werden sollen. Aus diesem Grund werden alle Actions der Cluster-MCTS zusammengeführt und auf mögliche Konflikte überprüft. Wird ein Konflikt identifiziert, z. B. wenn zwei FAUFs zur selben Zeit an einer Maschine bearbeitet werden sollen, wird dieser wie in Abbildung 4.14 gezeigt gelöst.

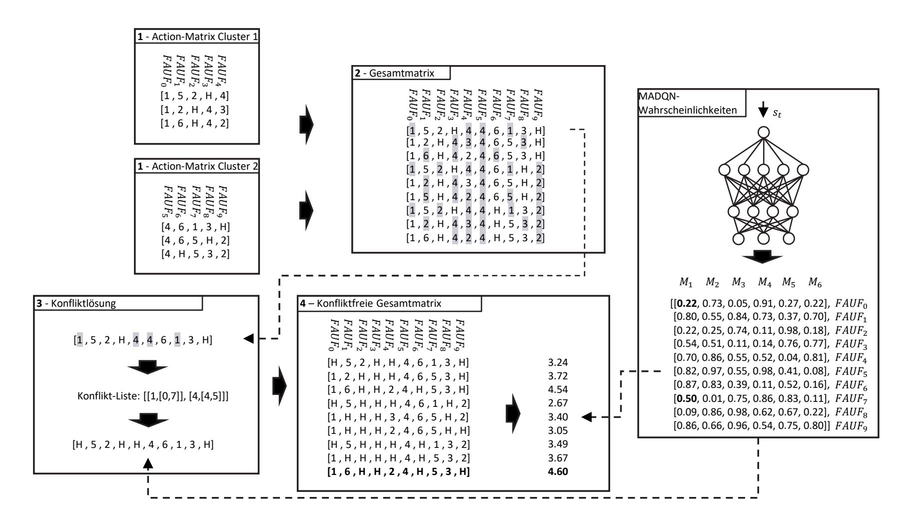


Abbildung 4.14: Action Zusammenführung

Im ersten Schritt werden in Abbildung 4.14 die Action-Matrizen der einzelnen Cluster-MCTS zu einer Gesamtmatrix zusammengeführt. Die daraus resultierende Gesamtmatrix beinhaltet alle FAUFs und alle gewählten Actions. Jede Zeile der Gesamtmatrix enthält Actions für alle FAUFs. Die grau hinterlegten Einträge sind Konflikte. So sollen in der ersten Zeile gleichzeitig zwei FAUFs jeweils an Maschine 1 und 4 bearbeitet werden. Diese zwei Konflikte werden im dritten Schritt gelöst. Dafür wird eine Konflikt-Liste erzeugt, in der die betroffene Maschine und die betroffenen FAUFs aufgelistet sind. Im Fall von Maschine 1 sind die betroffenen FAUFs 0 und 7, was einen Listeneintrag der Form [1,[0,7]] bedeutet. Für die beiden FAUFs 0 und 7 wird in der Wahrscheinlichkeits-Matrix des MADQNs die Eignungswahrscheinlichkeit an der Maschine 1 verglichen. Die

Wahrscheinlichkeits-Matrix des MADQNs beinhaltet alle Maschinen und FAUFs. Für jede Kombination aus Maschine und FAUF wird darin die Eignungswahrscheinlichkeit angegeben, ob die spezielle Kombination im Sinne des Optimierungsziels zu einer guten Gesamtlösung führen wird. Die Wahrscheinlichkeiten der FAUFs 0 und 7, an Maschine 1 ein gutes Gesamtergebnis herbeizuführen, sind unterschiedlich. Bei $FAUF_0$ beträgt die Wahrscheinlichkeit 22%, bei $FAUF_7$ beläuft sie sich auf 50%. Weil die Wahrscheinlichkeit, dass $FAUF_7$ an Maschine 0 zu einem guten Gesamtergebnis führt, höher ist wird $FAUF_7$ an Maschine 1 bearbeitet und $FAUF_0$ wird auf H wie ‚hold' (de: warten) gesetzt. Das bedeutet, dass $FAUF_0$ als zweites bearbeitet wird. Das Lösen der Konflikte geschieht analog für jede Zeile der Gesamtmatrix. Im letzten Schritt werden die einzelnen Zeilen der konfliktfreien Gesamtmatrix untereinander verglichen. Jede Zeile beinhaltet Actions für alle FAUFs und die Zeilen unterscheiden sich voneinander. Um herauszufinden, welche der Zeilen die beste Action-Kombination besitzt, werden wieder die MADQN Wahrscheinlichkeiten herangezogen. Die Wahrscheinlichkeiten der einzelnen Actions an den jeweiligen Maschinen werden aufaddiert. Die Zeile mit der höchsten Summe besitzt somit auch die höchste Wahrscheinlichkeit, ein gutes Gesamtergebnis herbeizuführen. Die so ausgewählten Actions können dann in die Fertigung zur Ausführung gegeben werden.

4.2.6 Trainingsframework zur Verbesserung der Generalisierungsfähigkeit

Wie in den Abschnitten 4.1.2 und 3.3 beschrieben, muss die Methode für die dynamische Terminierung über die Möglichkeit zur Generalisierung verfügen. Beim Training des MADQNs (vgl. Abschnitt 4.2.3) können aus zeitlichen Gründen nicht alle möglichen Störungsszenarien betrachtet werden. Die verschiedenen Störungsszenarien unterscheiden sich u. a. anhand folgender Eigenschaften:

- Störungszeitpunkt
- Störungsdauer

- Störungsanzahl
- betroffene Maschine
- Bearbeitungsfortschritt der einzelnen FAUFs
- aktuelle Position der einzelnen Bauteile in der Fertigung

Sollen alle potenziellen Störungsszenarien betrachtet werden, muss jede mögliche Kombination der Störungseigenschaften ermittelt und für das Training des MADQNs genutzt werden. Nachfolgend wird ein Trainingsframework vorgestellt, das die starke Generalisierungsfähigkeit (vgl. Abschnitt 2.4.4) des MADQNs ausnutzt, sodass nur ausgewählte Störungsszenarien beim Training berücksichtigt werden müssen. Die Lösungsqualität der beim Training nicht betrachteten Störungsszenarien soll dabei trotzdem maximiert werden. Eine grundlegende Unterscheidung der Störungsszenarien ist in Tabelle 4.1 dargestellt.

Tabelle 4.1: Einordnung der unterschiedlichen Störungsszenarien

		Gestörte Maschine			
		Maschine 1	Maschine 2	...	Maschine K
Störungs-zeitpunkt	t_0	Szenario 1,0	Szenario 2,0	...	Szenario K,0
	t_1	Szenario 1,1	Szenario 2,1	...	Szenario K,1
	t_2	Szenario 1,2	Szenario 2,2	...	Szenario K,2
	...	...	...	...	...
	t_T	Szenario 1,T	Szenario 2,T	...	Szenario K,T

Die Störungen treten darin zu verschiedenen Zeitpunkten $t \in \{0, 1, \ldots, T\}$ an verschiedenen Maschinen auf. Daraus ergibt sich ein Netzwerk aus Störungsszenarien, bei dem jedes einzelne Szenario eindeutig beschreibbar ist. Die Abstände der Störungszeitpunkte t können dabei frei gewählt werden. Beim Training des MADQNs werden die definierten Störungsszenarien mehrfach in einer zufälligen Reihenfolge durchlaufen. Das MADQN kann so eine Policy (vgl. Abschnitt 2.4.4) lernen, die alle die trainierten Szenarien nahezu optimal löst. Aufgrund

der Ähnlichkeit der verschiedenen Szenarien ist davon auszugehen, dass die Trainingszeit nicht linear mit der Anzahl an Störungsszenarien zunimmt. Somit wird sichergestellt, dass möglichst viele Störungsszenarien beim Training betrachtet werden können. Treten nach dem Training Störungen zu Zeitpunkten zwischen den trainierten Störungszeitpunkten auf, kann das MAQDN aufgrund seiner Generalisierungsfähigkeit auch zu diesen Zeitpunkten eine nahezu optimale Lösung finden.

Die die in Tabelle 4.1 dargestellten Störungsszenarien sind deterministisch; d. h., sowohl Störungsdauer als auch Störungszeitpunkt sind festgelegt und weichen beim Training nicht von den festgelegten Werten ab. Eine derart antrainierte Policy erzielt bei der dynamischen Terminierung die besten Ergebnisse, wenn die reale Störungsdauer und der reale Störungszeitpunkt mit den Trainingswerten übereinstimmen. In der Realität unterliegen diese Werte aber einer stochastischen Verteilung. Auch das kann beim Training des MADQNs berücksichtigt werden. Störungsszenarien, bei denen Maschinen ausfallen, die eine hohe Ausfallwahrscheinlichkeit besitzen, werden beim Training häufiger durchlaufen als Störungsszenarien von Maschinen mit einer geringen Ausfallwahrscheinlichkeit. Damit haben Störungsszenarien von Maschinen mit einer hohen Ausfallwahrscheinlichkeit einen stärkeren Einfluss auf die gelernte Policy, was in diesen Fällen zu besseren Terminierungsergebnissen führt. Auch die Störungsdauer unterliegt einer stochastischen Verteilung. Ausgehend von der durchschnittlichen Störungsdauer kann eine Standardabweichung festgelegt werden, sodass sich die Störungsszenarien auch in ihrer Störungsdauer unterscheiden.

Das beschriebene Training mit einer zufälligen Reihenfolge an Störungsszenarien kann grundsätzlich beliebig erweitert werden. So wäre es möglich, Störungen anhand ihrer Störungsdauer in verschiedene Klassen zu unterteilen. Diese Störungsklassen könnten beim Training zusätzlich berücksichtigt werden, um das Terminierungsergebnis zu verbessern. In der Validierung wird dieser Fall aber nicht aufgegriffen, da dafür die voraussichtliche Störungsdauer mit Eintritt der Störung bekannt sein muss. Die Störungsszenarien lassen sich grundsätzlich mit weiteren, anwendungsspezifischen Parametern anpassen, was jedoch die Trainingszeit

verlängert. Da sowohl die zur Verfügung stehende Trainingszeit als auch die Anzahl der anwendungsspezifischen Störungsparameter stark vom Anwendungsfall abhängen, kann hier kein allgemeingültiges Trainingsframework definiert werden. In der Validierung (vgl. Abschnitt 5) wird das Trainingsframework anhand der in diesem Kapitel beschriebenen Störungseigenschaften untersucht.

4.3 Zusammenfassung/Fazit

Die in diesem Kapitel entwickelte Methode soll die in Abschnitt 3.3 definierten Anforderungen einer Terminierung der flexiblen Fertigung erfüllen. Da in der Methode ein Multi-Agenten-Ansatz angewendet wird, ist auch eine Kombinationen der Actions der verschiedenen Agenten notwendig. Dafür wird die sogenannte ‚Action-Combination'-Methode eingeführt (vgl. Abschnitt 4.2.4). Um auch komplexe Optimierungsprobleme wie das OSSP lösen zu können, findet ein Clustering der FAUFs statt (vgl. Abschnitt 4.2.5). Dabei wird ein großes Optimierungsproblem mit vielen FAUFs in kleinere Sub-Probleme mit weniger FAUFs aufgeteilt, die leichter und besser zu lösen sind. Die vorgestellte Methode schließt damit die in Tabelle 3.3 definierte Forschungslücke vollumfänglich.

Das Zusammenwirken der verschiedenen Methoden verfolgt das Ziel, eine robuste dynamische Terminierung einer flexiblen Fertigung zu ermöglichen. Bei der Terminierung wird im Rahmen der vorliegenden Arbeit die Total Lead Time minimiert (vgl. Abschnitt 4.1.3). Die entwickelte Methode arbeitet domänenunabhängig und ist somit auch dazu in der Lage, andere Optimierungsziele zu verfolgen. So ist z. B. eine Optimierung des Energieverbrauchs bzw. eine Mischung aus Total Lead Time- und Energieverbrauchsminimierung möglich. Im nachfolgenden Kapitel wird die vorgestellte Methode validiert. Ob die Methode die Anforderungen an eine dynamische Terminierung erfüllt, wird dafür in verschiedenen Validierungsläufen geprüft.

5 Validierung

Nachfolgend wird die in Kapitel 4 entwickelte Methode anhand der in Abschnitt 3.3 definierten Anforderungen an eine dynamische Terminierung einer flexiblen Fertigung in verschiedenen Validierungsszenarien auf ihre Anwendbarkeit überprüft. Die Validierungen erfolgen sowohl in einem theoretischen, Open-Shop-Environment (vgl. Abschnitt 5.2) als auch in einer realen Fertigung (vgl. Abschnitt 5.3).

5.1 Definition der Validierungsumgebung

In diesem Abschnitt wird die Validierungsumgebung der in Kapitel 4 vorgestellten Methode definiert. Die entwickelte Methode macht sich die Funktionsprinzipien des Reinforcement Learnings zu nutze und interagiert mit einem Fertigungs-Environment, das in den folgenden Unterkapiteln erläutert wird.

5.1.1 FAUF-Agenten

Wie in Abschnitt 2.4.4 beschrieben, lernt ein Agent über die Interaktion mit einem Environment ein zielführendes Verhalten. Für die dynamische Terminierung wird jedem FAUF ein Agent zugewiesen; d. h., mehrere dieser FAUF-Agenten bewegen sich gleichzeitig in einem Fertigungs-Environment (vgl. Abschnitt 5.1.2) mit dem Ziel, ihre Fertigung so schnell wie möglich abzuschließen. Die FAUF-Agenten bekommen dafür einen Fertigungs-State (vgl. Abschnitte 2.4.4 und 5.1.2) und

entscheiden auf Grundlage dessen, welche Maschine sie für den nächsten Bearbeitungsschritt anfahren. Ein FAUF-Agent kann in dem Fertigungs-Environment verschiedene Zustände annehmen, wie im Zustandsdiagramm in Abbildung 5.1 dargestellt.

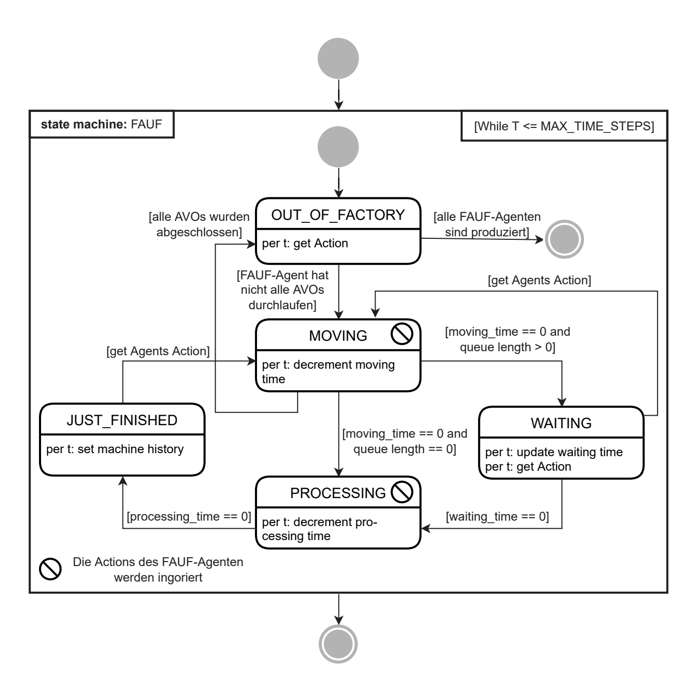


Abbildung 5.1: Zustandsdiagramm der Klasse ‚FAUF'

Bevor ein FAUF-Agent mit der Fertigung beginnen kann, ist dieser im Zustand *OUT_OF_Factory* (de: außerhalb der Fertigung). Ist die Fertigung für diesen FAUF-Agenten freigegeben, ermittelt dieser für jeden Zeitschritt t eine Action, in dem Fall eine Maschine, die er als nächste anfahren möchte. Nachdem der Agent sich für eine Maschine entschieden hat, ändert sich sein Zustand zu *MOVING* (de: in Bewegung), da das Bauteil nun zur ausgewählten Maschine transportiert wird. Die Transportzeit ist Bestandteil des Fertigungs-Environments (vgl. Abschnitt 5.1.2). Während des Transports kann sich der FAUF-Agent nicht für eine

andere Maschine entscheiden. Ist der Transport beendet und das Bauteil somit der entsprechenden Maschine übergeben worden, muss zwischen einer Maschine mit leerer bzw. ohne Warteschlange und Maschinen mit gefüllten Warteschlangen unterschieden werden. Ist eine Warteschlange vorhanden, ordnet sich der FAUF-Agent in diese ein. Er hat jetzt die Möglichkeit in jedem Zeitschritt t eine andere Maschine auszuwählen und die Warteschlange zu verlassen oder weiter zu warten. Ist die *waiting_time* (de: Wartezeit) $== 0$, findet die Fertigung des FAUFs statt - der FAUF-Agent wechselt in den Zustand *PROCESSING* (de: Bearbeitung). Während der Fertigung kann ein FAUF-Agent nicht zu einer anderen Maschine wechseln bzw. den Fertigungsprozess stoppen. Ist die Fertigung abgeschlossen, geht der FAUF-Agent in den Zustand *JUST_Finished* (de: gerade beendet) über und wählt von dort die nächste Action bzw. Maschine aus, zu der der FAUF-Agent hin transportiert werden muss. Wurden alle AVOs des FAUFs durchlaufen, wechselt der FAUF-Agent wieder in den Zustand *OUT_OF_Factory*.

5.1.2 Fertigungs-Environment

Wie in Abschnitt 2.4.4 dargelegt, ist die Interaktion eines Agenten mit einem Environment elementarer Bestandteil des RLs. Für die dynamische Terminierung wird ein Environment benötigt, das unter anderem die Maschinen sowie auch Arbeitspläne und Transport-, Rüst- und Fertigungszeiten abbilden kann (vgl. Abschnitt 4.1.2).

Prozessbedingte Zeiten

Prozessbedingte Zeiten sind Transport-, Rüst- und Fertigungszeiten. Diese Zeiten werden in Form von Matrizen innerhalb des Fertigungs-Environments beschrieben (vgl. Abbildung 5.2).

A) Transportzeit-Matrix

	M_1	M_2	M_3	M_4	M_5
M_1	[[0	3	2	4	6]
M_2	[3	0	3	2	4]
M_3	[2	3	0	5	4]
M_4	[4	2	5	0	2]
M_5	[6	4	4	2	0]]

B) Fertigungszeit-Matrix

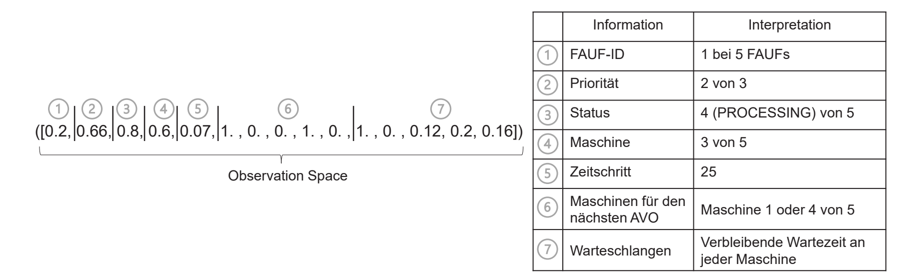
	$FAUF_1$	$FAUF_2$	$FAUF_3$	$FAUF_4$	$FAUF_5$
M_1	[[20	15	23	20	16]
M_2	[16	23	30	25	6]
M_3	[10	25	8	20	4]
M_4	[25	16	23	20	14]
M_5	[19	26	13	16	23]]

Abbildung 5.2: Beispiel: Prozessbedingte Zeiten in Matrixform

Die in Abbildung 5.2 dargestellten Matrizen zeigen ein Beispiel einer Transportzeit- und Fertigungszeit-Matrix in einem Fertigungs-Environment mit fünf Maschinen und fünf FAUF-Agenten bzw. Bauteilen. Die Rüstzeit kann als Bestandteil der Fertigungszeit angenommen werden, wenn die Rüstzeit nach jedem Bauteil anfällt. In diesem Fall ist eine separate Rüstzeit-Matrix nicht zwingend notwendig. Hängt die Rüstzeit von dem zuvor auf der Maschine gefertigten FAUF ab, ist eine Rüstzeit-Matrix erforderlich, aus der die Rüstzeitabhängigkeiten der verschiedenen FAUFs bzw. FAUF-Typen hervorgehen. Die prozessbedingten Zeiten werden abgefragt, wenn ein FAUF-Agent einen entsprechenden Zustand annimmt. Ist der Zustand des FAUF-Agenten z. B. *MOVING* (vgl. Abbildung 5.1), wird die Transportzeit aus der Transportzeit-Matrix entnommen und heruntergezählt (vgl. Abbildung 5.1.1).

Observation Space

Der State des Fertigungs-Environments setzt sich aus der Gesamtheit der Maschinenzustände, der Fertigungsfortschritte der FAUFs und der Warteschlangenbelegungen zusammen. Die einzelnen FAUF-Agenten bekommen nicht den gesamten State übermittelt, sondern einen sogenannten Observation Space (de: Beobachtungsraum), der für jeden FAUF-Agenten individuell ist. Der Observation Space besteht aus einem Array, das in sieben Bereiche unterteilt werden kann, bei denen jeder Eintrag einen Wert zwischen 0 und 1 annimmt (vgl. Abbildung 5.3).



	Information	Interpretation
①	FAUF-ID	1 bei 5 FAUFs
②	Priorität	2 von 3
③	Status	4 (PROCESSING) von 5
④	Maschine	3 von 5
⑤	Zeitschritt	25
⑥	Maschinen für den nächsten AVO	Maschine 1 oder 4 von 5
⑦	Warteschlangen	Verbleibende Wartezeit an jeder Maschine

Abbildung 5.3: Beispiel: Observation Space in einem Environment mit fünf FAUFs und fünf Maschinen

An erster Stelle des Observation Spaces in Abbildung 5.3 steht die FAUF-ID. Die FAUF-ID ist eindeutig und wird auf einen Wert zwischen 0 und 1 normalisiert. Bei fünf FAUFs werden die FAUF-IDs in Abständen von 0,2 bis 1 codiert. Als zweiter Eintrag folgt die FAUF-Priorität. Diese wird auch normalisiert, wobei der FAUF mit der höchsten Priorität den Wert 1 besitzt. Theoretisch können so beliebig viele unterschiedliche Prioritäten abgebildet werden. Im vorliegenden Beispiel hat der FAUF die Priorität 2 von 3. Der aktuelle Status des FAUFs wird mit dem dritten Eintrag übergeben. Bei fünf Maschinen entspricht 0,8 dem Zustand *PROCESSING* (vgl. Abbildung 5.1). Der vierte Eintrag beschreibt die aktuelle Maschine, an der sich der FAUF befindet. Der aktuelle Zeitschritt wird im fünften Eintrag des Observation Spaces abgebildet. An sechster Stelle folgt der sogenannte Action-Vektor. Dem FAUF-Agenten wird hiermit vorgegeben, welche Maschinen für den nächsten AVO zur Verfügung stehen. Maschinen, die mit einer 1 gekennzeichnet sind, können ausgewählt werden. Mit einer 0 markierte Maschinen stehen aufgrund einer Störung oder der Voraussetzungen des AVOs nicht zur Verfügung. Die siebte Information im Observation Space ist die Länge aller Warteschlangen. Diese Information spielt bei der Auswahl der nächsten Maschine eine wesentliche Rolle - grundsätzlich ist es das Ziel, die Total Lead Time (vgl. Abschnitt 4.1.3)zu minimieren, weshalb die FAUF-Agenten bevorzugt Maschinen mit einer niedrigeren Wartezeit anfahren.

Arbeitspläne der FAUFs

Die FAUF-Agenten können über ihre *FAUF_ID* eindeutig identifiziert werden und entscheiden aufgrund des Fertigungsstatus, zu welcher Maschine sie für den nächsten AVO transportiert werden möchten. Neben dem Fertigungsstatus ist bei der Maschinenauswahl auch der Arbeitsplan des entsprechenden Bauteils ausschlaggebend (vgl. Abschnitt 2.2.2). Der Arbeitsplan definiert die Abarbeitungsreihenfolge der AVOs und die für den jeweiligen AVO zur Verfügung stehenden Maschinen inklusive Rüst- und Fertigungszeiten. Innerhalb des Fertigungs-Environments wird der Arbeitsplan wie in Abbildung 5.4 dargestellt repräsentiert.

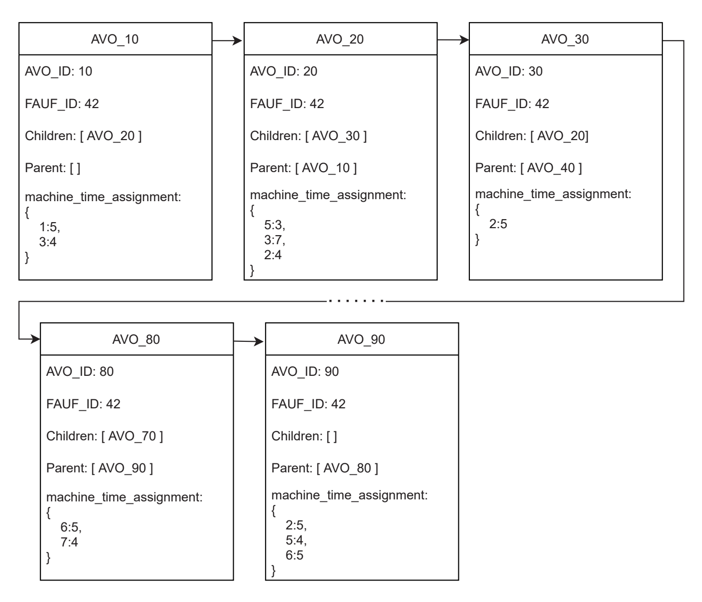


Abbildung 5.4: Beispiel: Repräsentation eines Arbeitsplans mit unterschiedlichen AVOs und alternativen Maschinen

Im Beispiel aus Abbildung 5.4 durchläuft ein FAUF mit der ID 42 die AVOs 10 bis 90. Jeder AVO hat eine eindeutige ID und beinhaltet Informationen über *Children* (de: Kinder), *Parents* (de: Eltern) und die für diesen AVO zur Verfügung stehenden Maschinen. Die *Children* definieren einen oder mehrere AVOs, die abgeschlossen sein müssen, um den aktuellen AVO auszuführen. AVOs, die auf den aktuellen AVO folgen, werden als *Parents* bezeichnet. Der erste AVO *(AVO_10)* besitzt keinen vorhergehenden AVO und analog dazu besitzt der letzte AVO *(AVO_90)* keinen Nachfolger. Da die AVOs in Abbildung 5.4 jeweils nur ein Child und ein Parent aufweisen, gibt es laut diesem Arbeitsplan keine Möglichkeit, die Abarbeitungsreihenfolge der AVOs zu ändern. Es besteht aber die Option mehrere Children und Parents zu definieren, woraus ein Netzwerk mit unterschiedlichen Abarbeitungsreihenfolgen resultiert (vgl. Abschnitt 2.2.3). Jedem AVO sind Maschinen und deren Fertigungszeiten als *machine_time_assignment* zugeordnet. Der *AVO_30* kann beispielsweise nur auf Maschine zwei gefertigt werden und benötigt fünf Zeitschritte für die Bearbeitung. Analog dazu können AVOs auch auf mehreren Maschinen abgearbeitet werden, die sich aber in ihrer Bearbeitungszeit unterscheiden.

5.2 Validierung der Methode an einem OSSP

Mit der in Kapitel 4 entwickelten Methode wird nachfolgend das OSSP gelöst. Dafür wird das für die Validierung genutzte OSSP zunächst definiert. Anschließend folgt die Validierung der Methode anhand verschiedener Kriterien und in unterschiedlichen Versuchsaufbauten.

5.2.1 Voraussetzungen für die OSSP-Validierung

Um das OSSP zu lösen, muss jedes Bauteil jede Maschine besucht haben (vgl. Abschnitt 2.3.1). Nachfolgende Validierungen werden, wenn nicht anders angegeben, in einem 4×10 Fertigungs-Environment mit $\mathcal{K}$ = vier Maschinen und $\mathcal{P}$ = zehn

FAUFs durchgeführt. Die Maschinen besitzen, wie in Tabelle 5.1 aufgeführt, voneinander abweichende Bearbeitungszeiten. Die Bearbeitungszeiten unterscheiden sich maximal um den Faktor zehn, sodass für diesen Fall kein sinnvoll taktgesteuerter Fertigungsprozess möglich ist. Zusätzlich werden in den nachfolgenden Validierungsbeispielen unterschiedliche Transportzeiten berücksichtigt (vgl. Tabelle 5.2). Die Rüstzeiten der Maschine werden als Teil der Bearbeitungszeit angesehen und sind deshalb nicht explizit angegeben.

Tabelle 5.1: Bearbeitungszeiten für ein Fertigungs-Environment mit $K = 4$

M0	M1	M2	M3
20	2	10	10

Tabelle 5.2: Transportzeiten für ein Fertigungs-Environment mit $K = 4$

	M0	M1	M2	M3
M0	1	2	3	4
M1	2	1	3	4
M2	3	3	1	4
M3	4	4	4	1

In der OSSP-Validierungsumgebung können mehrere FAUF-Agenten gleichzeitig oder versetzt die selbe Maschine für die Bearbeitung auswählen. In diesem Fall wird eine Warteschlange gebildet. Nachfolgende Validierungsszenarien im OSSP entsprechen somit der in Abschnitt 4.1.4 definierten Anforderung der *Maschinen mit Warteschlangen*. Zusätzlich wird davon ausgegangen, dass die Fertigung bzw. das Fertigungs-Environment zu Beginn komplett unbefüllt ist; d. h., dass die Warteschlangen der Maschinen leer sind und sich keine FAUFs in der Fertigung befinden. Ziel eines Fertigungsdurchlaufs ist es, eine entsprechende Anzahl an FAUF zu fertigen. Ein Fertigungsdurchlauf ist beendet, wenn alle FAUFs alle AVOs durchlaufen haben und fertig produziert sind. Der fest definierte Start- und End-Punkt eines Fertigungsdurchlaufs ermöglicht eine eindeutige und vergleichbare Lösungsqualität der Validierungsläufe und Methoden.

Als Optimierungsziel wird die Minimierung der Total Lead Time TLT bestimmt (vgl. Abschnitte 2.3.2 und 4.1.3). Dabei werden die Fertigungszeiten aller FAUFs addiert. Bei Validierungsläufen mit unterschiedlichen FAUF-Prioritäten wird die gewichtete Total Lead Time minimiert, bei der die Priorität des jeweiligen FAUFs mit der Anzahl der benötigten Fertigungszeitschritte multipliziert wird (vgl. Abschnitt 2.3.2).

5.2.2 OSSP-Validierungsläufe

Alle Validierungsläufe werden auf demselben Computer mit folgender Hardwarespezifikation ausgeführt: Intel(R) Core(TM)i7-9850H CPU @ 2.60GHz, 48959 MB Arbeitsspeicher. Für das Training der verschiedenen Validierungsmethoden wird kein GPU genutzt. Alle Methoden sind in Python 3.8 programmiert.

In den nachfolgenden Validierungsläufen werden unterschiedliche Methoden genutzt, um das OSSP zu lösen. Die Vergleichsmethoden werden der entwickelten Methode aus Kapitel 4 gegenübergestellt. Die entwickelte Methode wird nachfolgend als $MCTS + \theta_m$ bezeichnet, da sie zum einen auf einer *MCTS* und zum anderen auf dem Multi-Agenten-Ansatz des MADQNs beruht (vgl. Abschnitt 4.2.1), das durch seine Netzwerkparameter θ_m repräsentiert wird (vgl. Kapitel 2.4.5). Eine reine, nicht unterstütze MCTS wird nachfolgend als *MCTS* bezeichnet (vgl. Abschnitt 2.5.2). Der Vergleich von *MCTS* und $MCTS+\theta_m$ ermöglicht eine Aussage darüber, ob die Unterstützung der MCTS durch das MADQN zu besseren Ergebnissen führt. Zusätzlich werden auch die Ergebnisse des MADQNs θ_m für den Vergleich herangezogen.

Da die entwickelte Methode auf einem Multi-Agent DQN beruht erfolgt die Validierung analog dazu an einem Ansatz, der einen Single-Agent verwendet. Dafür wird ein Single-Agent DQN genutzt (vgl. Kapitel 2.4.5), das nachfolgend als θ_s bezeichnet wird. Die Kombination dieses Single-Agent DQNs mit einer MCTS wird als $MCTS+\theta_s$ definiert. In Ergänzung dazu werden auch zwei komplett andere Ansätze, in Form von Heuristiken (vgl. Abschnitt 3.2.2), für die Validierung eingesetzt. Eine Heuristik soll die Arbeitsweise eines Mitarbeiters in der Fertigung

abbilden. Um die Transportwege so gering wie möglich zu halten, bringt dieser Mitarbeiter ein FAUF immer zur nächstmöglichen freien Maschine. Diese Heuristik wird als *Fair Distribution* (de: gleichmäßige Verteilung) [82] bezeichnet und in den nachfolgenden Abbildungen als *Fair* gekennzeichnet. Die zweite Heuristik ist die Metaheuristik *Partikelschwarmoptimierung* [68], die nachfolgend als *PSO* bezeichnet wird. Die *PSO* kann das OSSP unter bestimmten Kriterien optimal lösen.

Da das Optimierungsziel die Minimierung der Total Lead Time (TLT) ist (vgl. Abschnitt 4.1.3), werden zunächst die TLTs der zuvor vorgestellten Lösungen in Fertigungs-Environments mit unterschiedlicher Größe verglichen. Die Größe des Fertigungs-Environments wird in der Form $\mathcal{K} \times \mathcal{N}$ angegeben, wobei K für die Anzahl der Maschinen und N für die Anzahl der FAUFs steht. Die Total Lead Times der ausgewählten Methoden in den verschieden großen Fertigungs-Environments sind in Abbildung 5.5 dargestellt.

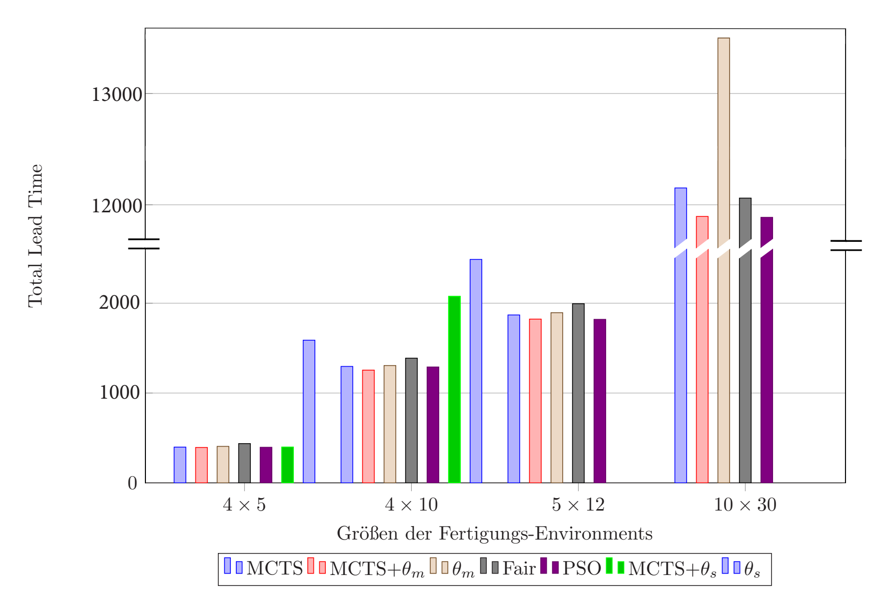


Abbildung 5.5: Total Lead Time in verschiedenen Fertigungs-Environments mit einem fixen Verzweigungsfaktor $|\mathcal{A}| = 5$.

Die Methoden, die auf künstlicher Intelligenz beruhen, müssen im ersten Schritt angelernt werden. Hierzu wird jede dieser Methoden für jedes Fertigungs-Environment in unter 10 Stunden angelernt. Für die Ansätze, die auf einer MCTS beruhen, wird der Durchschnitt über 50 Simulations-Durchläufe gebildet. Die Anzahl der Rollouts einer Simulation ist festgelegt auf $20 \cdot |\mathcal{A}|$. Der Verzweigungsfaktor $|\mathcal{A}|$ ist bei allen Durchläufen konstant 5. Die Total Lead Times aller verglichenen Methoden nehmen mit einer wachsenden Größe des Environments zu. Für eine bessere Vergleichbarkeit sind die TLTs und deren relative Abweichung von der besten TLT des jeweiligen Environments in Tabelle 5.3 dargestellt.

Tabelle 5.3: Vergleich der Total Lead Time und der durchschnittlichen Abweichung in verschiedenen Fertigungs-Environments im Open Shop

	4×5		4×10		5×12		10×30		
Methode	**TLT** $[t]$	**Abweichung** [%]	**TLT** $[t]$	**Abweichung** [%]	**TLT** $[t]$	**Abweichung** [%]	**TLT** $[t]$	**Abweichung** [%]	**∅ Abweichung** [%]
MCTS	398	1,015	1296	3,267	1869	2,692	12151	2,221	2,299
MCTS+θ_m	**394**	**0**	**1255**	**0**	1823	0,165	11895	0,067	**0,058**
θ_m	406	3,046	1306	4,064	1894	4,066	13499	13,561	6,184
Fair	437	10,914	1388	10,598	1994	9,560	12060	1,455	8,132
PSO	396	0,508	1290	2,789	**1829**	**0**	**11887**	**0**	0,824
MCTS+θ_s	399	1,269	2075	65,339					
θ_s	1588	303,046	2487	98,167					

Anmerkung: $|\mathcal{A}|$ = 5 für alle Läufe mit einer MCTS

Im 4×5 und 4×10 Fertigungs-Environment erreicht die MCTS+θ_m die geringste Total Lead Time (vgl. Tabelle 5.3). In beiden Environments ist damit die unterstützte MCTS+θ_m besser als die nicht unterstützte MCTS. Sowohl MADQN θ_m als auch die MCTS kommen einzeln für sich genommen auf schlechtere Total Lead Times als die Kombination aus beiden. Die PSO-Metaheuristik erzielt mit einer relativen Abweichung von 0,508% für das 4×5 Environment und 2,789% für das 4×10 Environment die zweitniedrigste TLT. Das Single-Agent DQN θ_s erreicht im 4×5 Environment nur eine sehr schlechte Total Lead Time, die eine Abweichung von über 300% zur geringsten TLT aufweist. Für das 5×12 und das 10×30 Environment liegen keine Ergebnisse von θ_s vor, da der Single-Agent diese Environments mit der gegebenen Hardware bzw. innerhalb der vorgegebenen Trainingszeit nicht erlernen kann. Analog dazu liegen auch keine Ergebnisse von MCTS+θ_s vor. In den größeren Environments 5×12 und 10×30 erzielt die PSO die geringste Total Lead Time. Die MCTS+θ_m weicht von der besten TLT im 5×12 und 10×30 Environment nur 0,165% bzw. 0,067% ab. Auch in diesen Environments ist die entwickelte Methode MCTS+θ_m besser als eine nicht unterstützte MCTS bzw. ein θ_m. In der letzten Spalte ist die durchschnittliche relative Abweichung zur geringsten TLT über alle Environments abgebildet. Die MCTS+θ_m minimiert die Total Lead Time über alle Environments am besten. Die zweitbeste Methode ist die PSO, deren durchschnittliche Abweichung aber um einen Faktor von über zehn schlechter als jene des MCTS+θ_m ist.

Zusammenfassend lässt sich festhalten, dass ein Single-Agent größere Environments aufgrund deren Komplexität schlecht oder gar nicht erlernen kann. Der Einsatz eines Multi-Agenten-Systems für die Anwendung zur dynamischen Terminierung erscheint daher sinnvoll. Die Kombination aus MCTS und MADQN (MCTS+θ_m) ist in jedem untersuchten Environment besser als eine nicht unterstützte MCTS bzw. das Multi-Agent θ_m. Die in Kapitel 4 entwickelte Methode zur Minimierung der Total Lead Time verbessert somit nachweislich die Ergebnisqualität.

Für die Validierungsläufe in Tabelle 5.3 wird ein fester Verzweigungsfaktor von $|\mathcal{A}| = 5$ angenommen. Nachfolgend wird der Einfluss des Verzweigungsfaktors auf die Total Lead Time untersucht (vgl. Abbildung 5.6). Dafür wird eine reine MCTS mit der MCTS+θ_m in einem 4×10 Environment verglichen. Das in Abbildung 5.6 eingezeichnete MADQN θ_m entspricht dem θ_m, das die MCTS+θ_m unterstützt. Die Anzahl der Iterationen pro Zeitschritt ist auch hier $20 \cdot |\mathcal{A}|$.

Der Verzweigungsfaktor $|\mathcal{A}|$ gibt an, wie viele neue Knoten während der aktuellen MCTS-Iteration untersucht werden. Jeder Knoten entspricht dabei einer Action, z. B. ein FAUF-Agent entscheidet sich für die nächste Maschine. Je mehr Knoten untersucht werden, desto höher ist die Chance, einen Knoten zu finden, der die Total Lead Time minimiert. Mit der Anzahl der zu untersuchenden Knoten nimmt aber auch die Rechenzeit der MCTS zu, sodass zwischen der Ergebnisqualität und der Rechenzeit abgewogen werden muss. Aus diesem Grund wird in Abbildung 5.6 die erreichte Total Lead Time über verschiedene Verzweigungsfaktoren veranschaulicht. Dabei wird die durchschnittliche TLT und deren Streuung in 50 Durchläufen dargestellt.

Wie in Tabelle 5.3 dargestellt, ist die Total Lead Time der MCTS+θ_m geringer als jene der reinen MCTS. Zudem konvergiert die MCTS+θ_m deutlich schneller gegen eine minimale TLT. Besonders groß ist der Unterschied bei $|\mathcal{A}| = 5$. Obwohl die MCTS und die MCTS+θ_m die gleiche Anzahl an Knoten untersuchen, differiert sich die durchschnittliche TLT um mehr als 2% (vgl. Tabelle 5.3). Das lässt sich damit begründen, dass die MCTS die Knoten zufällig auswählt. Aus diesem Grund ist auch die Streuung der Total Lead Times der MCTS viel größer. Die MCTS+θ_m hingegen nutzt das MADQN θ_m bei der Auswahl des Knotens, der expandiert bzw. untersucht wird (vgl. Abschnitt 4.2.2).

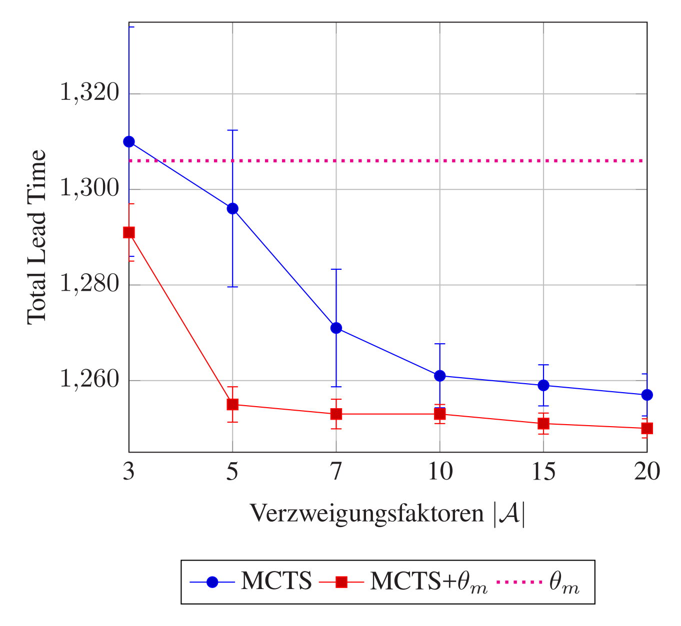


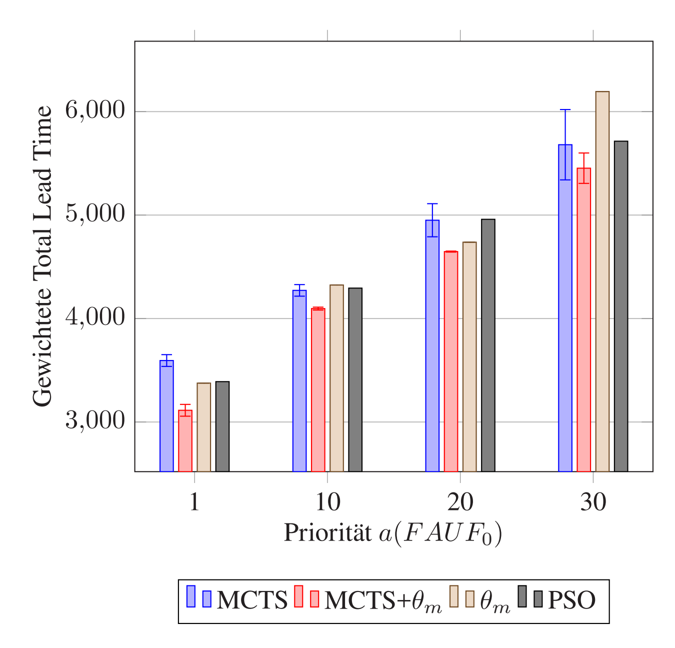
Abbildung 5.6: Vergleich der Total Lead Time über verschiedene Verzweigungsfaktoren $|\mathcal{A}|$, $K = 4$, $N = 10$

Mit Blick auf die in Abschnitt 3.3 definierten Anforderungen an eine Methode für die dynamische Terminierung lässt sich für dieses Beispiel als Analyseergebnis festhalten, dass die in Kapitel 4 entwickelte Methode schneller und robuster als eine nicht unterstützte MCTS ist. Schon mit einem Verzweigungsfaktor von 5

konvergiert die MCTS+θ_m gegen eine minimale Total Lead Time (vgl. Abbildung 5.6). Damit werden weniger Rollouts für das gleiche TLT-Ergebnis benötigt als bei der nicht unterstützten MCTS, was die Berechnungszeit verkürzt. Zusätzlich ist die MCTS+θ_m robuster, was an den eingezeichneten Minima und Maxima zu erkennen ist. Die durchschnittliche Streuung für die Verzweigungsfaktoren 3, 5, 7 beträgt für die MCTS 17,6 und für die MCTS+θ_m 4,5. Die Streuung der Ergebnisse der MCTS+θ_m ist damit deutlich geringer, sodass im vorliegenden Beispiel eine Total Lead Time mit einer Streuung von vier Zeitschritten als Abweichung zum Durchschnitt sichergestellt werden kann.

Da es in der Fertigung FAUFs mit unterschiedlichen Prioritäten geben kann, muss eine Methode für die dynamische Terminierung auch mit verschiedenen FAUF-Prioritäten umgehen können. Dafür werden in Abbildung 5.7 und Tabelle 5.4 eine nicht unterstützte MCTS, die MCTS+θ_m, das θ_m und die PSO verglichen.

Das Fertigungs-Environment für die Validierungsläufe in Abbildung 5.7 besteht aus vier Maschinen und zehn FAUFs. Die Prioritäten a der FAUFs 1 bis 9 bleiben während der Versuchsläufe unverändert. Die Priorität $a(FAUF_0)$ nimmt in den unterschiedlichen Läufen Werte von 1, 10, 20 und 30 an. Bei Versuchsläufen mit Prioritäten wird die gewichtete Total Lead Time angegeben, bei der jeder Zeitschritt mit der entsprechenden FAUF-Priorität multipliziert wird (vgl. Abschnitt 2.3.2). FAUFs mit einer hohen Priorität fallen damit pro Zeitschritt viel stärker ins Gewicht als solche mit einer geringen. Um die gewichtete TLT zu minimieren, muss eine Lösungsmethode dazu in der Lage sein die Prioritäten zu berücksichtigen und FAUFs mit einer hohen Priorität entsprechend schneller fertigzustellen (vgl. Abschnitt 4.1.2).



Prioritäten: a($FAUF_1$)=1, a($FAUF_2$)= 10, a($FAUF_3$)= 1, a($FAUF_4$)= 10, a($FAUF_5$)= 1, a($FAUF_6$)= 10, a($FAUF_7$)= 1, a($FAUF_8$)= 1, a($FAUF_9$) $= 1$

Abbildung 5.7: Vergleich der gewichteten Total Lead Times für unterschiedliche Prioritäten von $FAUF_0$ mit $|\mathcal{A}| = 20, N = 4, M = 10$

In Abbildung 5.7 und Tabelle 5.4 ist zu erkennen, dass die gewichtete TLT mit der Zunahme der Priorität von $FAUF_0$ wie erwartet ansteigt. Die TLT der MCTS+θ_m ist im Schnitt $6,308\%$ besser als die der PSO. In Verbindung mit den Ergebnissen aus Tabelle 5.3 ist damit nachgewiesen, dass die MCTS+θ_m im Vergleich mit den anderen untersuchten Lösungsmethoden nicht nur die geringste Total Lead Time erreicht, sondern auch im Falle sehr stark abweichender FAUF-Prioritäten die geringste gewichtete TLT ermöglicht.

Tabelle 5.4: Gewichtete Total Lead Time im Open-Shop-Environment für unterschiedliche Prioritäten für $a(FAUF_0)$ mit $|\mathcal{A}| = 20, N = 4, M = 8$

Methode	$a(FAUF_0) = 1$ TLT $[t]$	$a(FAUF_0) = 1$ Abweichung [%]	$a(FAUF_0) = 10$ TLT $[t]$	$a(FAUF_0) = 10$ Abweichung [%]	$a(FAUF_0) = 20$ TLT $[t]$	$a(FAUF_0) = 20$ Abweichung [%]	$a(FAUF_0) = 30$ TLT $[t]$	$a(FAUF_0) = 30$ Abweichung [%]	∅ Abweichung [%]
MCTS	3594	15,451	4272	4,297	4959	6,520	5680	4,163	7,608
MCTS+θ_m	**3113**	**0**	**4096**	**0**	**4647**	**0**	**5453**	**0**	**0**
θ_m	3376	8,448	4324	5,566	4738	1,958	6194	13,589	7,390
PSO	3390	8,898	4294	4,834	4959	6,714	5714	4,786	6,308

Prioritäten: a($FAUF_1$)=1, a($FAUF_2$)= 10, a($FAUF_3$)= 1, a($FAUF_4$)= 10, a($FAUF_5$)= 1, a($FAUF_6$)= 10, a($FAUF_7$)= 1, a($FAUF_8$)= 1, a($FAUF_9$) $= 1$

Da in der Fertigung Störungen auftreten, müssen Methoden, die für die dynamische Terminierung genutzt werden sollen, auch in diesen Fällen dazu in der Lage sein, die TLT zu minimieren. In Abbildung 5.8 und Tabelle 5.5 wird die Total Lead Time unterschiedlicher Methoden für den Fall einer Maschinenstörung verglichen. Die Störungsdauer MDT (vgl. Abschnitt 2.2.5) ist in allen Läufen auf 50 Zeitschritte festgesetzt und tritt für jeden Durchlauf nur an einer Maschine auf.

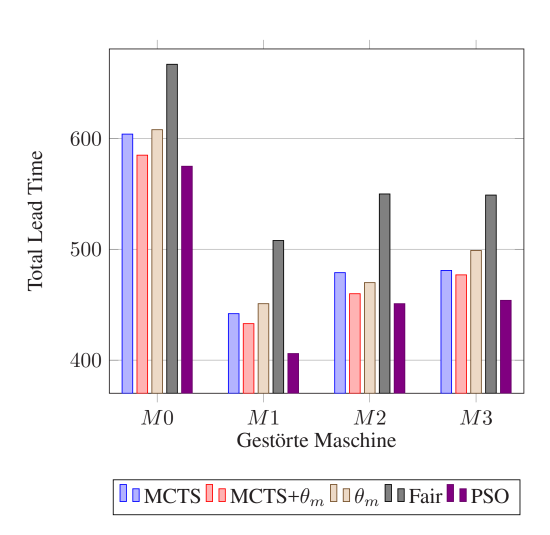


Abbildung 5.8: Total Lead Time für unterschiedliche Maschinen mit Störungen - ohne Störungen beim Training zu berücksichtigen. $MDT = 50$, K = 4, N = 5, $|\mathcal{A}|$ = 5

Das θ_m, das auch für die MCTS+θ_m verwendet wird, wird für die Beispiele in Abbildung 5.8 innerhalb 100 Tsd. Iterationen bzw. Trainingsdurchläufen angelernt. Die Trainingsdauer beträgt dafür ca. fünf Minuten. Bei diesem Training durchläuft das θ_m keine Szenarien, bei denen Störungen auftreten. Den Fall einer Störung bzw. der beschränkten Verfügbarkeit von Fertigungs-Ressourcen berücksichtigt das θ_m beim Training nicht. Die Ergebnisse in Abbildung 5.8 bestätigen die vorhergehenden Validierungsläufe, bei denen die MCTS+θ_m die TLT stets besser minimierte als eine nicht unterstützte MCTS bzw. das θ_m. Die PSO Metaheuristik kann Störungsszenarien abdecken und minimiert die Total Lead Time so am besten. Im Durchschnitt ist in diesem Versuch die Total Lead Time der PSO 3,865% besser als jene der MCTS+θ_m (vgl. Tabelle 5.5). Aufgrund der Generalisierungsfähigkeit des θ_m ist die MCTS+θ_m aber dennoch in der Lage, ein im Training nicht betrachtetes Fertigungsszenario relativ gut zu lösen. Das θ_m und analog dazu auch die MCTS+θ_m erfüllen somit das Kriterium der starken Generalisierungsfähigkeit (vgl. Abschnitt 2.4.4).

Tabelle 5.5: Vergleich der Total Lead Times im Falle unterschiedlicher Maschinenstörungen. MDT=50, K=4, N=5, $|\mathcal{A}|$ = 5

	Gestörte Maschine 0		**Gestörte Maschine 1**		**Gestörte Maschine 2**		**Gestörte Maschine 3**		
Methode	**TLT** $[t]$	**Abweichung** [%]	**TLT** $[t]$	**Abweichung** [%]	**TLT** $[t]$	**Abweichung** [%]	**TLT** $[t]$	**Abweichung** [%]	**∅ Abweichung** [%]
MCTS	604	5,043	442	8,867	479	6,682	481	5,947	6,635
MCTS+θ_m	585	1,739	433	6,650	460	2,450	477	5,066	3,976
θ_m	608	5,739	451	11,084	470	4,677	499	9,912	7,853
MCTS+θ_m mit Trainingsframework	**575**	**0**	410	0,985	**449**	**0**	455	0,220	0,301
θ_m mit Trainingsframework	601	4,522	437	7,635	463	3,118	489	7,709	5,746
Fair	667	16,000	508	25,123	550	22,494	549	20,925	21,163
PSO	**575**	**0**	**406**	**0**	451	0,445	**454**	**0**	**0,111**

Beim Training von MCTS+θ_m und θ_m werden keine Störungen berücksichtigt. Beim Einsatz eines Trainingsframeworks für MCTS+θ_m und θ_m werden Störungen berücksichtigt.

Um die Lösungsqualität der MCTS+θ_m zu verbessern, durchläuft das θ_m im nächsten Schritt beim Training die verschiedenen Störungsszenarien (in Tabelle 5.5 durch den Zusatz ‚mit Trainingsframework‘ gekennzeichnet). Anstatt der 100

Tsd. Iterationen im vorhergehenden Test werden jetzt insgesamt 400 Tsd. Iterationen durchlaufen, wobei jedes einzelne Störungsszenario, bei dem eine Maschine ausfällt, insgesamt 100 Tsd. Mal durchlaufen wird. Dadurch vervierfacht sich auch die Berechnungszeit von 5 auf 20 Minuten. Die Ergebnisse dieses Validierungslaufes sind in Abbildung 5.9 dargestellt.

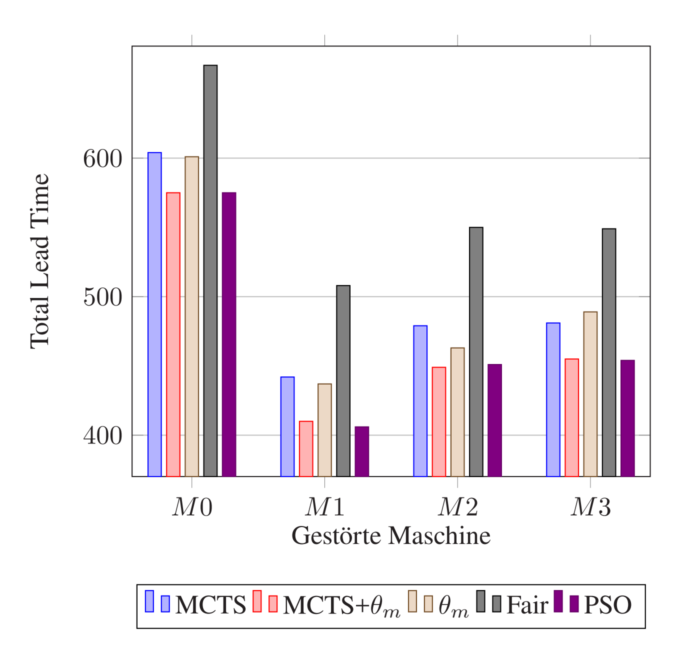


Abbildung 5.9: Total Lead Time für unterschiedliche Maschinen mit Störungen - Störungen werden beim Training berücksichtigt. $MDT = 50$, $K = 4$, $N = 5$, $|\mathcal{A}| = 5$

Die Ergebnisse von MCTS, Fair und PSO bleiben unverändert, da diese keinem Trainingsprozess unterliegen. Das MADQN θ_m verringert, aufgrund der Einbeziehung von Störungsszenarien im Trainingsprozess, die durchschnittliche Abweichung zur Lösungsqualität der PSO von 3,976% auf 0,301%. Analog dazu verbessert sich auch die Total Lead Time der MCTS+θ_m. Die MCTS+θ_m ist damit in dem getesteten Szenario im Durchschnitt um 0,19% schlechter als die PSO. Werden Störungen beim Training des θ_m berücksichtigt, kann die entwickelte Methode MCTS+θ_m somit ähnliche Total Lead Times wie eine Metaheuristik

erzielen. Gleiches lässt sich auch in größeren Fertigungs-Environments reproduzieren (vgl. Abbildung 5.10).

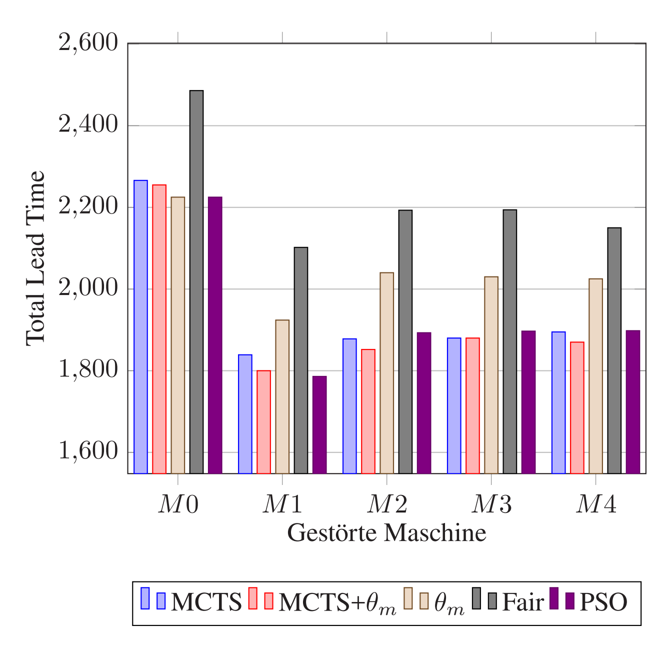


Abbildung 5.10: Total Lead Time für unterschiedliche Maschinen mit Störungen - Störungen werden beim Training berücksichtigt. $MDT = 50$, $K = 5$, $N = 12$, $|\mathcal{A}| = 5$

In Abbildung 5.10 sind die Total Lead Times aller untersuchten Methoden in einem 5 $\times$12 Fertigungs-Environment dargestellt. Die mit Störungen angelernte MCTS+θ_m erzielt in diesem Fertigungs-Environment eine TLT, die im Schnitt um 0,49% besser als die der PSO ist. Die Validierungsläufe in Abbildung 5.8, 5.9 und 5.10 zeigen zum einen, dass die MCTS+θ_m das Kriterium der starken Generalisierungsfähigkeit erfüllt, und zum anderen, dass gezieltes Antrainieren von Störungsszenarien einen deutlichen Anstieg der Lösungsqualität in den Störungsszenarien bewirkt. Aufgrund der Vielzahl an möglichen Störungsszenarien in einer realen Fertigung können diese aus zeitlichen Gründen nicht alle beim Training betrachtet werden. Mit Hilfe eines Trainingsframeworks, wie in Abschnitt 4.2.6 vorgestellt,

wird im nachfolgenden Versuch überprüft, ob die starke Generalisierungsfähigkeit dazu genutzt werden kann, nur einige ausgewählte Störungsszenarien anzulernen und damit trotzdem eine gute Lösungsqualität für alle Störungsszenarien sicherzustellen (vgl. Abbildung 5.11). Dafür wird die Total Lead Time über den Anteil an angelernten Störungsszenarien abgebildet. Die Trainingszeit bleibt für jeden Versuch gleich, nur die Anzahl der Störungsszenarien variiert beim Training.

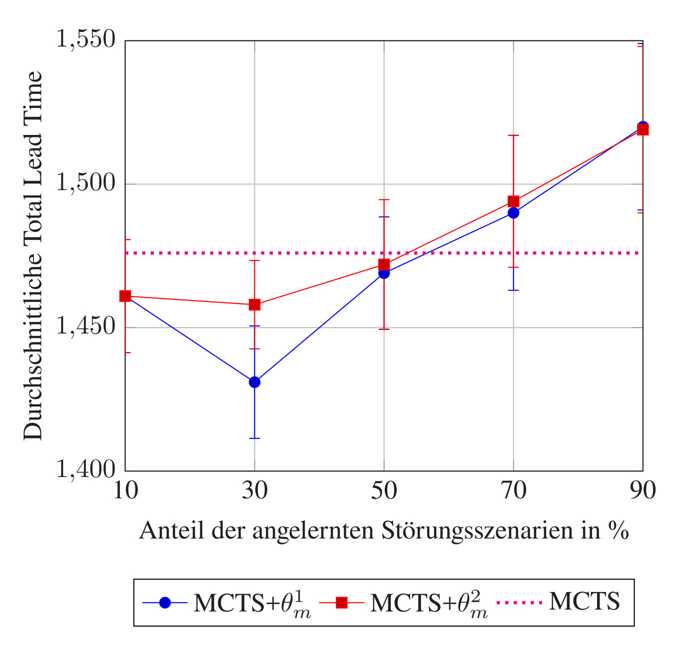


Abbildung 5.11: Durchschnittliche Total Lead Time über den Anteil der angelernten Störungsszenarien, $K = 4$, $N = 10$, $|\mathcal{A}| = 10$, $MDT = 50$

Verglichen werden in Abbildung 5.11 zwei unterstützte MCTS, MCTS+θ_m^1 und MCTS+θ_m^2, mit einer nicht unterstützten MCTS. Die beiden unterstützten MCTS unterscheiden sich bei der Auswahl der Störungsszenarien für das Training. Wie in Abschnitt 5.3.1 definiert, beginnt ein Fertigungsdurchlauf mit einer leeren Fertigung, in der entsprechend viele Bauteile gefertigt werden müssen. Der Fertigungsdurchlauf ist beendet, wenn alle Bauteile alle AVOs durchlaufen haben.

Die MCTS+θ_m^1 wählt für das Training gleichzeitig Störungsszenarien aus, die zu Beginn und zum Ende eines Fertigungsdurchlaufs auftreten. Der Startzeitpunkt einer Störung wird innerhalb der unterschiedlichen Trainingsläufe variiert, wobei die Störungsdauer MDT unverändert bleibt. Die MCTS+θ_m^2 wählt die Störungsszenarien, beginnend mit dem Start des Fertigungsdurchlaufs, schrittweise aus. Störungen, die erst zum Ende eines Fertigungsablaufs auftreten, werden bei dieser Methode nur dann betrachtet, wenn der Anteil an Störungsszenarien relativ hoch ist.Sowohl die MCTS+θ_m^1 als auch die MCTS+θ_m^2 erreichen bei einem Anteil von 30% an angelernten Störungsszenarien die geringste durchschnittliche Total Lead Time (vgl. Abbildung 5.11). Dabei ist die MCTS+θ_m^1 um 1,9% besser als die MCTS+θ_m^2. Mit einem wachsenden Anteil an Störungsszenarien gleichen sich die durchschnittlichen Total Lead Times beider unterstützten MCTS an. Zusätzlich steigt die TLT beider unterstützten MCTS mit einer Zunahme an Störungsszenarien.

Die geringere durchschnittliche Total Lead Time der MCTS+θ_m^1 in Abbildung 5.11 lässt sich mit der Auswahl der Störungsszenarien begründen. Im Gegensatz zur MCTS+θ_m^2 werden bei der MCTS+θ_m^1 sowohl Störungsszenarien zu Beginn als auch zum Ende eines Fertigungsdurchlaufes für das Training genutzt. Mit einem Anteil an angelernten Störungsszenarien von 30%, bei denen sich die einzelnen Störungsszenarien stark voneinander unterscheiden, ist die MCTS+θ_m^1 aufgrund der starken Generalisierungsfähigkeit dazu in der Lage, unbekannte, leicht abweichende Störungsszenarien zu lösen. Die für das Anlernen der MCTS+θ_m^2 genutzten Störungsszenarien sind infolge der Szenarienauswahl sehr ähnlich, sodass eine Generalisierung auf gänzlich andere Störungsszenarien, z. B. Störungen, die zum Ende eines Fertigungsdurchlaufes auftreten, nicht möglich ist. Sowohl bei der MCTS+θ_m^1 als auch bei der MCTS+θ_m^2 verschlechtert sich die durchschnittliche TLT mit einer Zunahme an angelernten Störungsszenarien (vgl. Abbildung 5.11). Das ist darauf zurückzuführen, dass mit einer Erweiterung der Trainingsszenarien auch die Trainingszeit bzw. die Trainingsepisoden zunehmen müssen. Für die Praxis bedeutet das, dass ein höherer Anteil an Störungsszenarien im

Training zu exponentiell steigenden Trainingszeiten führt. Die starke Generalisierungsfähigkeit der MCTS+θ_m ermöglicht es aber, mit einer gezielten Auswahl von Störungsszenarien für das Training in jedem Fall gute Total Lead Times zu erreichen (vgl. Tabelle 5.5).

Die Validierungsläufe in einem OSSP-Environment zeigen, dass die MCTS+θ_m dazu in der Lage ist, das OSSP besser zu lösen als eine nicht unterstützte MCTS oder eine Metaheuristik (vgl. Tabelle 5.3). Die MCTS+θ_m besitzt in den durchgeführten Versuchen eine leicht bessere Lösungsqualität als die Metaheuristik. Im Gegensatz zur Metaheuristik ist die Funktionsweise der MCTS+θ_m domainunabhängig, was bedeutet, dass Änderungen der Fertigung, z. B. die Einbindung neuer Maschinen oder neue Arbeitspläne, nur dem Fertigungs-Environment übergeben werden müssen. Während des Trainings lernt die MCTS+θ_m, die neuen Gegebenheiten sinnvoll umzusetzen, um das Optimierungsziel zu erfüllen, also die Total Lead Time zu minimieren. Bei einer Metaheuristik muss hingegen das mathematische Modell angepasst werden, was eine sehr komplexe Aufgabe darstellt. Auch in dem Fall, dass FAUFs mit mehreren, unterschiedlichen Prioritäten betrachtet werden müssen, erreicht die MCTS+θ_m die geringste TLT aller Vergleichsmethoden (vgl. Abbildung 5.7). Um auch im Falle von Störungen eine minimale TLT zu ermöglichen, wird in der Validierung ein Trainingsframework untersucht, das die starke Generalisierungsfähigkeit der MCTS+θ_m ausnutzt, um nicht alle Störungsszenarien anlernen zu müssen, aber dennoch eine gute Lösungsqualität in unbekannten Störungsszenarien herbeiführt (vgl. Abbildung 5.11).

Die in Kapitel 4 konzipierten Erweiterungen zur klassischen MCTS führen dazu, dass die MCTS+θ_m in allen Versuchen besser als eine nicht unterstützte MCTS bzw. ein Multi-Agent θ_m ist (vgl. Tabelle 5.3). Die Wirksamkeit der entwickelten Methode MCTS+θ_m ist damit bestätigt.

5.3 Validierung der Methode an einem realen Anwendungsfall

Die in Kapitel 4 vorgestellte Methode wird nachfolgend an einem realen Beispiel einer Stanzerei validiert. Dafür werden zunächst die Voraussetzungen und Gegebenheiten der Stanzerei definiert. Anschließend folgen unterschiedliche Validierungsläufe, um die Anwendbarkeit der entwickelten Methode für die dynamische Terminierung einer realen Fertigung zu untersuchen.

5.3.1 Voraussetzungen der Stanzerei-Fertigung

Die Stanzerei, in der die Validierung erfolgen soll, stellt Rotoren und Statoren für Elektromotoren her. Der Rotor dreht sich innerhalb des Stators und dessen Magnetfelds und wandelt so elektrische in mechanische Energie um. Sowohl die Rotoren als auch die Statoren bestehen aus einzelnen Metall-Blechen, die im Anschluss an den Stanzprozess untrennbar miteinander verbunden werden. Ausgangsmaterial für die jeweiligen Metall-Bleche sind Metall-Coils (vgl. Abbildung 5.12).



Abbildung 5.12: Metall-Coils - Ausgangsmaterial für die Rotor- und Stator-Bleche

Die Metall-Coils unterscheiden sich in Größe, Dicke und der Art der Metall-Legierung. Im ersten Schritt findet das sogenannte *Vorstanzen* statt. Dabei werden aus dem Metall-Coil zunächst in zwei Schritten die sogenannten Ausfallronden gestanzt (vgl. Abbildung 5.13).

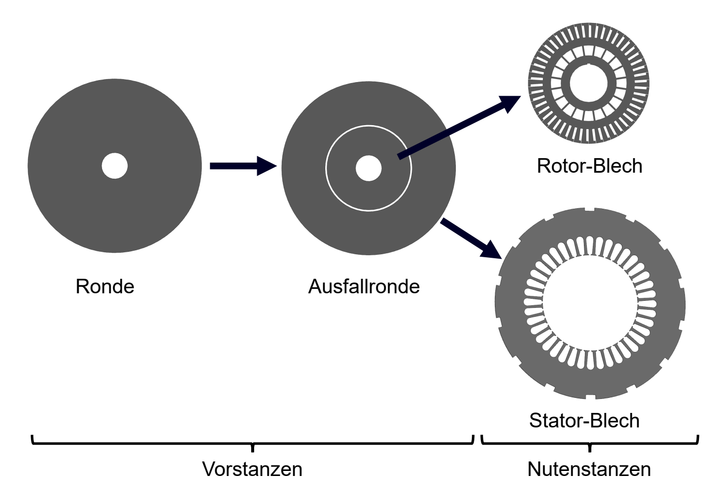


Abbildung 5.13: Zwischen- und End-Produkte der verschiedenen Stanzprozesse

In den Abbildungen 5.14 und 5.15 werden Rotor-Blechpakete vor und nach dem Nutenstanzen gezeigt.



Abbildung 5.14: Rotor-Blechpaket vor dem Nutenstanzen



Abbildung 5.15: Rotor-Blechpaket nach dem Nutenstanzen

In Abbildung 5.16 ist die Einspannung eines Rotor-Bleches in der Nutenstanze dargestellt. Das Stanz-Werkzeug stanzt pro Hub eine Nute aus. Anschließend dreht sich die Einspannung samt Rotor-Blech, um beim nächsten Hub die danebenliegende Nute auszustanzen. Die Nutenstanzen unterscheiden sich untereinander im Automatisierungsgrad. Die in Abbildung 5.16 gezeigte Nutenstanze bearbeitet die Rotor-Bleche vollautomatisch. Andere Nutenstanzen benötigen manuelle Unterstützung beim Stanzen, was zu unterschiedlichen Bearbeitungszeiten führt.

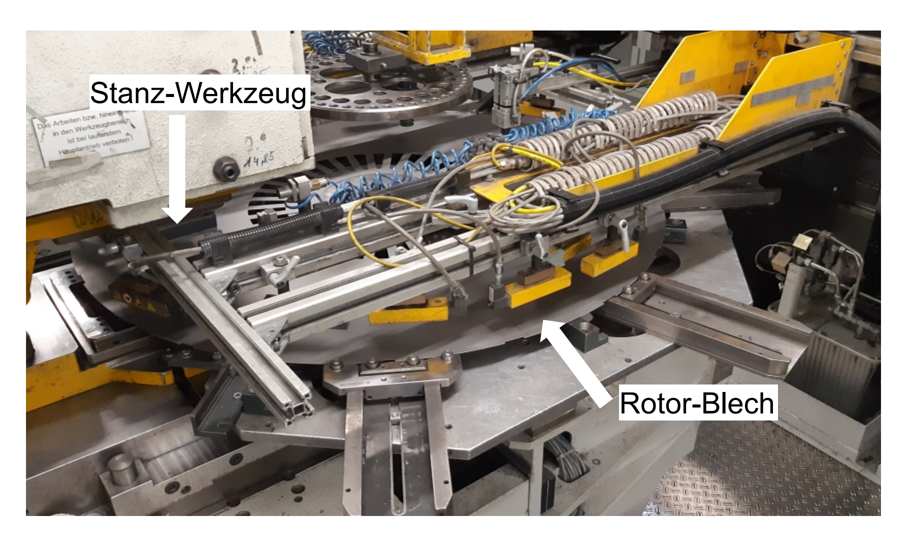


Abbildung 5.16: Eingespanntes Rotor-Blech in einer Nutenstanze

Die Stanzerei besteht aus drei Vorstanzen und 15 Nutenstanzen. Aufgrund der langen Vorlaufzeiten (bestehend aus Stanz-Werkzeugbereitstellungs- und Rüstzeiten) der Vorstanzen eignen sich diese nicht für eine dynamische Terminierung, die eine schnelle Reaktion auf Planabweichungen in der Fertigung ermöglicht. Aus diesem Grund beschränken sich die nachfolgenden Untersuchungen auf eine dynamische Terminierung für die 15 Nutenstanzen. Die Arbeitspläne beinhalten alternative Maschinen und die Vorlaufzeit für das Nutenstanzen ist deutlich geringer als jene für das Vorstanzen. Im Folgenden wird angenommen, dass für das Nutenstanzen eines jeden Bleches fünf AVOs notwendig sind, wobei für jeden AVO mehrere Nutenstanzen zur Verfügung stehen. Für die Validierung werden vier verschiedene Arbeitspläne der FAUFs berücksichtigt, die sich bei der Maschinenauswahl, der Bearbeitungszeit und der Anzahl an alternativen Maschinen je AVO unterscheiden. Die Arbeitspläne sind im Anhang beschrieben (vgl. Kapitel A.2). Aufgrund der Anzahl der FAUFs und der möglichen alternativen Bearbeitungsmaschinen ergeben sich in dem Stanzerei-Environment schon bei acht FAUFs, wobei jeder Arbeitsplan zwei Mal vertreten ist, insgesamt $2,177 \cdot 10^9$ mögliche Fertigungsabläufe. Bei 16 FAUFs, jeder Arbeitsplan wird vier Mal durchlaufen, resultieren

hieraus $4,738 \cdot 10^{18}$ mögliche Fertigungsabläufe. Die Stanzerei wird deswegen als FJSSP definiert (vgl. Abschnitt 2.3.1). Die Komplexität ist somit entsprechend hoch und kann von keinem Menschen beherrscht werden.

Für die dynamische Terminierung werden Informationen aus verschiedenen IT-Systemen und zu unterschiedlichen Zeitpunkten benötigt (vgl. Abbildung 5.17). Zum einen sind *statische Informationen* aus dem ERP- und APS-System erforderlich (vgl. Abschnitt 4.1.2). Diese Informationen umfassen z. B. Arbeitspläne, FAUF-Informationen, die Transportzeit-Matrix, aber auch den terminierten Produktionsplan. In der Regel liegen alle diese Informationen vor Beginn der Fertigung vor und ändern sich während des Fertigungsprozesses nicht. Aus diesem Grund können diese Informationen als statisch eingeordnet werden.

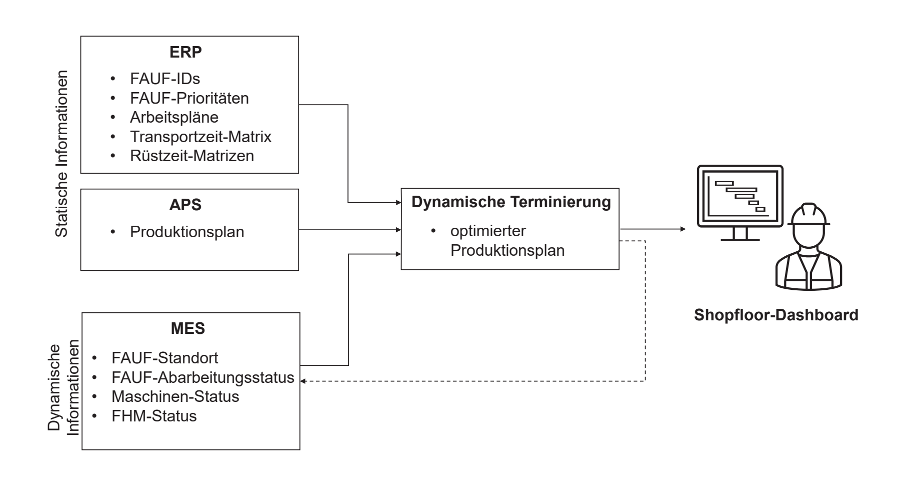


Abbildung 5.17: Möglicher Informationsfluss für die dynamische Terminierung

Zum anderen gibt es *dynamische Informationen* (vgl. Abschnitt 4.1.2). Diese Informationen können sich über die Zeit permanent ändern und beinhalten z. B. den Abarbeitungsstatus der FAUFs oder den jeweiligen Maschinen-Status. Anhand der dynamischen Informationen kann eine Störung erkannt und die dynamische Terminierung angestoßen werden. Das Ergebnis der dynamischen Terminierung

kann den entsprechenden Mitarbeitern entweder über ein sogenanntes Shopfloor-Dashboard mitgeteilt bzw. visualisiert oder direkt an das MES weitergeleitet werden, von wo aus dann die geeigneten Maßnahmen eingeleitet werden.

5.3.2 Stanzerei-Validierungsläufe

Nachfolgend werden verschiedene Validierungsläufe im Stanzerei-Environment durchgeführt, bei denen jeweils die MCTS, die MCTS+θ_m, das θ_m und die PSO verglichen werden. Analog zu den Validierungsläufen im Open Shop (vgl. Abschnitt 5.2) wird auch in denjenigen der Stanzerei die Total Lead Time bzw. die gewichtete TLT minimiert (vgl. Abschnitt 2.3.2). Im ersten Validierungslauf werden die TLTs im Stanzerei-Environment für unterschiedliche Anzahlen an FAUFs verglichen (vgl. Abbildung 5.18).

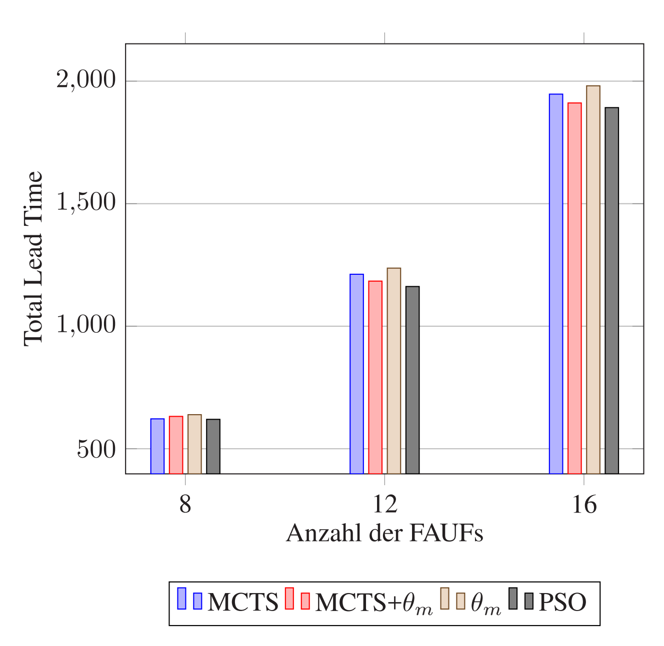


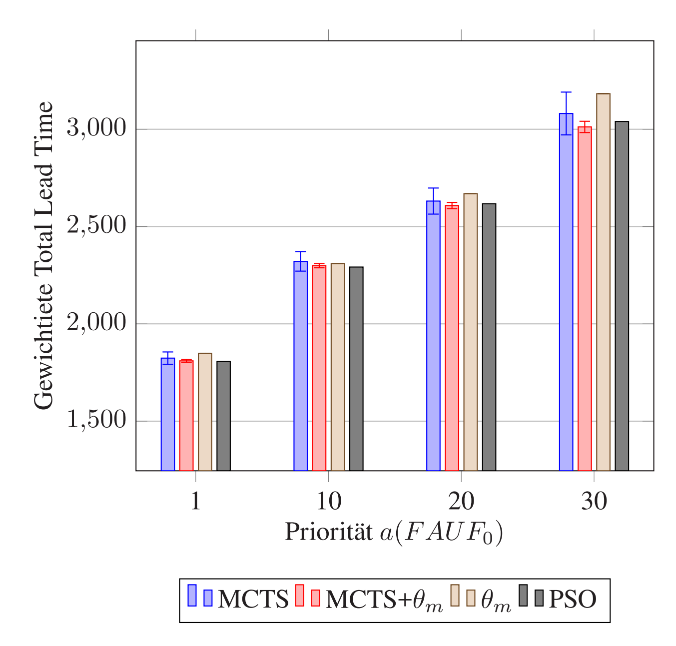
Abbildung 5.18: Vergleich der Total Lead Time in Stanzerei-Environments mit einer unterschiedlichen Anzahl an FAUFs, $N = 12$, $|\mathcal{A}| = 5$

Um den Ablauf einer Fertigung möglichst exakt abzubilden, starten nicht alle FAUFs gleichzeitig mit der Bearbeitung. Alle zwei Zeitschritte werden vier FAUFs dem Stanzerei-Environment übergeben, bis die maximale Anzahl an FAUFs für das entsprechende Stanzerei-Environment erreicht ist. Wie in Abbildung 5.18 zu sehen ist, minimiert die PSO die Total Lead Time in den untersuchten Szenarien am besten. Die genauen Werte und deren prozentuale Abweichung zur jeweils besten TLT sind in Tabelle 5.6 aufgeführt.

Tabelle 5.6: Total Lead Time und prozentuale Abweichung in verschiedenen Stanzerei-Environments mit $|\mathcal{A}| = 5$

	8 FAUFs		**12 FAUFs**		**16 FAUFs**		
Methode	**TLT** $[t]$	**Abweichung** $[\%]$	**TLT** $[t]$	**Abweichung** $[\%]$	**TLT** $[t]$	**Abweichung** $[\%]$	**∅ Abweichung** $[\%]$
MCTS	622	0,323	1212	4,303	1947	2,907	2,511
MCTS+θ_m	632	1,935	1184	1,893	1911	1,004	1,611
θ_m	639	3,065	1237	6,454	1981	4,704	4,741
PSO	**620**	**0**	**1162**	**0**	**1892**	**0**	**0**

Die durchschnittliche Abweichung der TLT von der MCTS+θ_m zur PSO beträgt 2,511% (vgl. Tabelle 5.6). Im Vergleich dazu erreicht die MCTS+θ_m in der OSSP-Validierung geringere TLTs als die PSO. Das OSSP- und das Stanzerei-Environment unterscheiden sich in erster Linie durch die Einführung von Arbeitsplänen und die Tatsache, dass nicht alle FAUFs an jeder Maschine bearbeitet werden müssen. Da die FAUFs der Stanzerei unterschiedliche Prioritäten besitzen können, wird nachfolgend die gewichtete TLT (vgl. Formel 2.4) für variierende FAUF-Prioritäten untersucht. Dafür wird die Priorität von $FAUF_0$ im 12×8 Stanzerei-Environment schrittweise von eins auf 30 erhöht, wobei die gewichteten TLTs der einzelnen Methoden verglichen werden (vgl. Abbildung 5.19).



Prioritäten: a($FAUF_1$)=1, a($FAUF_2$)= 10, a($FAUF_3$)= 1, a($FAUF_4$)= 10, a($FAUF_5$)= 1, a($FAUF_6$)= 10, a($FAUF_7$)= 1

Abbildung 5.19: Gewichtete Total Lead Time im Stanzerei-Environment für unterschiedlich Prioritäten für $a(FAUF_0)$. $|\mathcal{A}| = 20, K = 8$

Die gewichtete TLT eines FAUFs berechnet sich als Summer aller Zeitschritte t, die der FAUF für die Fertigung benötigt, multipliziert mit der FAUF-Priorität a (vgl. Abschnitt 2.3.2). Hohe Werte für die Priorität a wirken sich somit sehr stark auf die gewichtete Total Lead Time eines FAUFs aus. Ziel ist es daher, FAUFs mit einer hohen Priorität schneller abzuschließen, um die gewichtete TLT der gesamten Fertigung zu minimieren.

Aufgrund der Steigerung der Priorität $a(FAUF_0)$ über die verschiedenen Validierungsläufe in Abbildung 5.19 nimmt auch die TLT zu. Die exakten Werte der Validierungsläufe sind in Tabelle 5.7 aufgelistet.

Tabelle 5.7: Gewichtete Total Lead Time im Stanzerei-Environment für unterschiedliche Prioritäten für $a(FAUF_0)$. $|\mathcal{A}| = 20, M = 8,$

	$a(FAUF_0) = 1$		$a(FAUF_0) = 10$		$a(FAUF_0) = 20$		$a(FAUF_0) = 30$		
Methode	**TLT** [t]	**Abweichung** [%]	**TLT** [t]	**Abweichung** [%]	**TLT** [t]	**Abweichung** [%]	**TLT** [t]	**Abweichung** [%]	**∅ Abweichung** [%]
MCTS	1824	0,941	2321	1,265	2631	0,882	3081	2,291	1,345
MCTS+θ_m	1810	0,166	2299	0,305	**2608**	**0**	**3012**	**0**	**0,118**
θ_m	1849	2,324	2310	0,785	2669	2,339	3183	5,677	2,781
PSO	**1807**	**0**	**2292**	**0**	2617	0,345	3040	0,930	0,319

Prioritäten: a($FAUF_1$)=1, a($FAUF_2$)=10, a($FAUF_3$)=1, a($FAUF_4$)=10, a($FAUF_5$)=1, a($FAUF_6$)=10, a($FAUF_7$)=1

Die geringste in Tabelle 5.7 dargestellte durchschnittliche Abweichung unter den verglichenen Validierungsmethoden erreicht die MCTS+θ_m mit 0,118%. Auch hier zeigt sich, dass die Kombination aus MCTS und θ_m zu besseren Ergebnissen führt als die einzelnen Methoden für sich allein. Die PSO erzielt in den Validierungsszenarien eine Abweichung von 0,319% (vgl. Tabelle 5.7). Damit bestätigen sich die Resultate der Validierungsläufe im OSSP für unterschiedliche Prioritäten, auch wenn die Differenz zwischen den durchschnittlichen Abweichungen von MCTS+θ_m und PSO kleiner wird (vgl. Abbildung 5.7).

Der nächste Validierungslauf untersucht die Generalisierungsfähigkeit der verschiedenen Methoden im Stanzerei-Environment. Dafür werden die MCTS+θ_m und das θ_m zunächst in einem Stanzerei-Environment ohne Störungen angelernt. Die Validierungsläufe erfolgen anschließend in einem Stanzerei-Environment, bei dem pro Durchlauf zufällig eine Maschine ausfällt, mit einer MDT von 150 Zeitschritten (vgl. Abschnitt 2.2.5). Eine Methode, die auf KI basiert, benötigt zum Lösen dieser Aufgabe eine starke Generalisierungsfähigkeit (vgl. Abschnitt 2.4.4). Die Validierungsergebnisse dieses Versuches sind in Abbildung 5.20 und Tabelle 5.8 dargestellt.

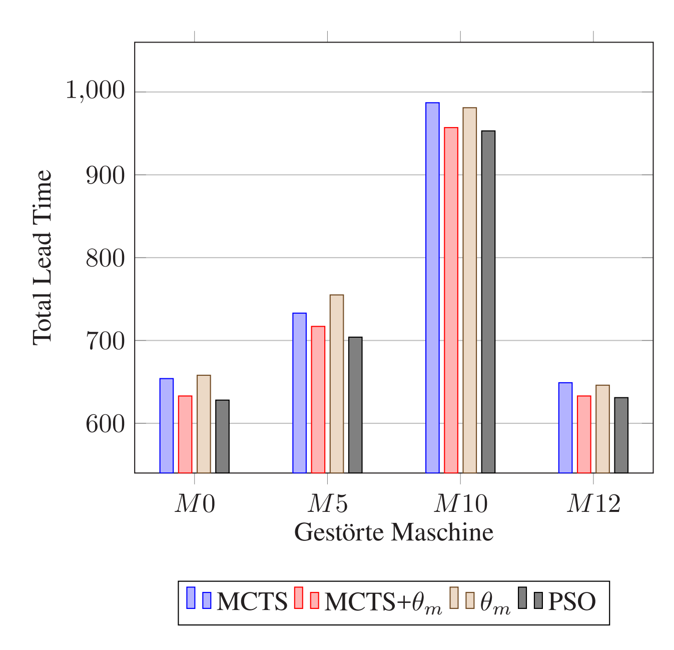


Abbildung 5.20: Total Lead Time im Stanzerei-Environment für unterschiedliche Maschinen mit Störungen - Störungen werden beim Training nicht berücksichtigt. $MDT = 150$, $K = 15$, $N = 8$, $|\mathcal{A}| = 10$

In allen untersuchten Störungsszenarien ist die PSO besser als die MCTS+θ_m. Die TLTs der MCTS+θ_m sind im Durchschnitt um 0,845% schlechter als jene der PSO (vgl. Tabelle 5.8). Dieser Versuch bestätigt die starke Generalisierungsfähigkeit der MCTS+θ_m. Obwohl Störungen beim Training der KI-Methode nicht berücksichtigt wurden, ist die MCTS+θ_m dazu in der Lage, die Szenarien zu lösen, und ist dabei nur wenig schlechter als die PSO. Wird ein Trainingsframework wie in Abbildung 5.10 verwendet, kann die Lösungsqualität weiter gesteigert werden. Damit ist die MCTS+θ_m dazu imstande, bei angelernten Szenarien (bzw. Szenarien, die angelernten Szenarien sehr ähnlich sind) eine vergleichbare Minimierung der Total Lead Time zu erreichen wie die PSO. Zusätzlich kann die MCTS+θ_m aufgrund ihrer starken Generalisierungsfähigkeit auch komplett unbekannte Störungs-Szenarien lösen.

Tabelle 5.8: Total Lead Time im Stanzerei-Environment für unterschiedliche Maschinen mit Störungen - Störungen werden beim Training nicht berücksichtigt. $MDT = 150$, $K = 15$, $N = 8$, $|\mathcal{A}| = 10$

	Gestörte Maschine 0		**Gestörte Maschine 5**		**Gestörte Maschine 10**		**Gestörte Maschine 12**		
Methode	**TLT** $[t]$	**Abweichung** [%]	**TLT** $[t]$	**Abweichung** [%]	**TLT** $[t]$	**Abweichung** [%]	**TLT** $[t]$	**Abweichung** [%]	**∅ Abweichung** [%]
MCTS	654	4,140	733	4,119	987	3,568	649	2,853	3,670
MCTS+θ_m	633	0,796	717	1,847	957	0,420	633	0,317	0,845
θ_m	658	4,777	755	7,244	981	2,938	646	2,377	4,334
PSO	**628**	**0**	**704**	**0**	**953**	**0**	**631**	**0**	**0**

Um eine Skalierbarkeit der MCTS+θ_m sicherzustellen, wird in Kapitel 4.2.5 ein Clustering der FAUFs entwickelt. Die FAUFs werden dabei verschiedenen Clustern zugeordnet. FAUFs innerhalb eines Clusters sind sich sehr ähnlich, d. h. sie müssen für die verbleibenden AVOs die gleichen Maschinen nutzen. Jedes Cluster löst das Scheduling-Problem mittels einer eigenen MCTS+θ_m (vgl. Kapitel 4.2.5). Die Scheduling-Ergebnisse der verschiedenen Cluster werden anschließend kombiniert und an die Fertigung zur Abarbeitung gegeben. Dieser Ansatz ermöglicht eine theoretisch unendliche Skalierbarkeit der Methode, die in der Realität aufgrund von Hardwarebegrenzungen beschränkt ist. Nachfolgend wird, analog zu der Validierung in Abbildung 5.18, anhand des Stanzerei-Environments die Auswirkung des Clusterns auf die Total Lead Time dargestellt (vgl. Abbildung 5.21).

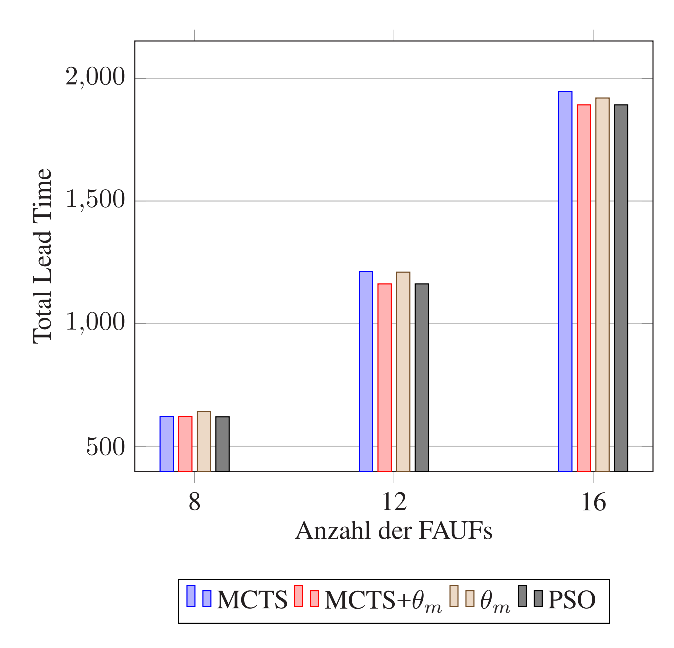


Abbildung 5.21: Total Lead Time im Stanzerei-Environment über verschiedene FAUF-Anzahlen. Die MCTS+θ_m clustert FAUFs. $K = 15$, $|\mathcal{A}| = 5$

In Tabelle 5.9 sind die TLTs und die Abweichung zur jeweiligen geringsten TLT aufgelistet. Die PSO erreicht in allen Szenarien die geringste TLT. Die MCTS+θ_m erreicht in den Stanzerei-Environments mit 12 und 16 FAUFs die gleiche TLT wie die PSO und weicht nur im ersten Stanzerei-Environment mit 8 FAUFs leicht ab, sodass sich eine durchschnittliche Abweichung von 0,108% ergibt.

Tabelle 5.9: Total Lead Time im Stanzerei-Environment über verschiedene FAUF-Anzahlen. Die MCTS+θ_m clustert FAUFs. $M = 15$, $|\mathcal{A}|$ = 5

	8 FAUFs		**12 FAUFs**		**16 FAUFs**		
Methode	**TLT** $[t]$	**Abweichung** $[\%]$	**TLT** $[t]$	**Abweichung** $[\%]$	**TLT** $[t]$	**Abweichung** $[\%]$	**∅ Abweichung** $[\%]$
MCTS	622	0,323	1212	4,303	1947	2,907	2,511
MCTS+θ_m	622	0,323	**1162**	**0**	**1892**	**0**	0,108
θ_m	641	3,387	1210	4,131	1920	1,480	2,999
PSO	**628**	**0**	**1162**	**0**	**1892**	**0**	**0**

Die durchschnittliche Abweichung der MCTS+θ_m von 0,108% liegt deutlich unter derjenigen der Validierungsläufe ohne das Clustern, die 2,511% beträgt (vgl. Tabelle 5.6). Die Verbesserung der durchschnittlichen Abweichung durch das Clustern lässt sich damit erklären, dass durch das Clustering der FAUFs der Lösungsraum in Sub-Lösungsräume aufgeteilt wird (vgl. Abschnitt 4.2.5). Die MCTS+θ_m kann den Sub-Lösungsraum bei gleichbleibender Berechnungsdauer effizienter durchsuchen. Die Wahrscheinlichkeit, ein optimales bzw. ein sehr gutes Ergebnis für das Sub-Problem zu finden, ist folglich viel größer, sodass sich letztlich die Lösungsqualität verbessert. Aufgrund der vorliegenden Validierungsergebnisse kann davon ausgegangen werden, dass das Clustering inklusive aller Teilfunktionen wirksam ist und somit die Lösungsqualität verbessert sowie die Skalierbarkeit der in Kapitel 4 entwickelten Methode sicherstellt.

Die im Stanzerei-Environment durchgeführten Validierungsläufe zeigen, dass die Metaheuristik PSO in allen Versuchen minimal bessere Ergebnisse herbeiführt als die MCTS+θ_m. Generell kann die PSO dieses Optimierungsproblem also besser lösen.

5.4 Diskussion der Validierungsergebnisse

Um die Einsatzmöglichkeit der in Kapitel 4 entwickelten Methode MCTS+θ_m für die dynamische Terminierung einer flexiblen Fertigung zu untersuchen, werden im vorliegenden Kapitel 5 verschiedene Validierungsläufe unternommen. Alle Validierungsläufe haben das Ziel, die MCTS+θ_m dahingehend zu überprüfen, ob die Methode die in Abschnitt 3.3 definierten Anforderungen an eine dynamische Terminierung erfüllt.

Für die Validierungsläufe im Open Shop (vgl. Abschnitt 5.2) lässt sich festhalten, dass die MCTS+θ_m in allen Versuchen zu wesentlich besseren Ergebnissen als die MCTS oder das θ_m führt (vgl. Tabelle 5.3). Die entwickelte Methode erreicht eine bessere Total Lead Time, was in der Fertigung mit geringeren Durchlaufzeiten gleichzusetzen ist. Sogar im Vergleich mit einer Metaheuristik, die das OSSP sehr gut lösen kann, zeigt die MCTS+θ_m eine bessere TLT (vgl. Tabelle 5.3). Die MCTS+θ_m ist somit dazu in der Lage, das OSSP sehr gut zu lösen bzw. trotz der **Komplexität** des Optimierungsproblems eine gute Lösung zu finden. Eine besondere Stärke der MCTS+θ_m lässt sich beobachten, wenn FAUF-Prioritäten bei der Terminierung berücksichtigt werden (vgl. Tabelle 5.4). Ohne zusätzliche Anpassungen bzw. Erweiterungen kann die MCTS+θ_m das Problem mit Abstand am besten lösen. Die MCTS+θ_m arbeitet dabei domainunabhängig. Änderungen des Fertigungs-Environments, z.B. andere FAUF-Prioritäten, neue Maschinen oder neue Arbeitspläne, führen nicht zu Änderungen der MCTS+θ_m. Die MCTS+θ_m minimiert die Total Lead Time in jedem Fertigungs-Environment und beachtet dabei die im Modell eingeführten Randbedingungen (vgl. Abschnitt 5.1.2). Die MCTS+θ_m erfüllt demnach die Anforderung aus Abschnitt 3.3, ein so komplexes Optimierungsproblem lösen zu können. Zusätzlich kann die Anforderung einer **Action-Combination**, weil ein Multi-Agenten-System genutzt wird, implizit als erfüllt angesehen werden, da die Optimierungsergebnisse der MCTS+θ_m ohne eine Action-Combination nicht erreichbar wären.

Neben der Anforderung, auch komplexe Optimierungsprobleme lösen zu können, ist ebenso die **Robustheit** der Lösung von besonderer Bedeutung (vgl. Abschnitt 3.3). Die Validierungsergebnisse der Versuche zum Verzweigungsfaktor (vgl. Abschnitt 4.2.4 und Abbildung 5.6) und der Total Lead Time für unterschiedliche Prioritäten (vgl. Abbildung 5.7) ermöglichen einen Rückschluss auf die Streuung der Ergebnisse von jeweils 20 Durchläufen. Die Streuung der MCTS ist dabei immer viel größer als jene der MCTS+θ_m. Das lässt sich damit erklären, dass die MCTS neue Unter-Konten zufällig auswählt bzw. exploriert (vgl. Abschnitt 2.5.2). Auch wenn die MCTS den Entscheidungsbaum asynchron aufbaut und während der Selektionsphase den Knoten mit der höchsten Wahrscheinlichkeit für ein gutes Endergebnis aussucht, findet die Unter-Knoten-Auswahl zufällig statt und beeinflusst somit das Ergebnis des MCTS-Durchlaufs stark. Die MCTS+θ_m wählt die Unter-Knoten mit Hilfe des θ_m aus (vgl. Abschnitt 4.2.3). Die Selektion der Unter-Knoten beruht folglich nicht auf dem Zufall, was die Robustheit der MCTS+θ_m erklärt. Die MCTS+θ_m erfüllt demnach die Anforderung an eine robuste Lösungsqualität.

Eine weitere in Abschnitt 3.3 definierte Anforderung ist die **Generalisierungsfähigkeit**. Diese wird vorausgesetzt, da nicht alle Störungsszenarien beim Training betrachtet werden können. In Tabelle 5.5 wird die Total Lead Times für verschiedene Störungsszenarien verglichen. Dabei ist zu erkennen, dass die TLT der MCTS+θ_m deutlich besser als jene der MCTS und des θ_m ist. Das lässt auf eine starke Generalisierungsfähigkeit der MCTS+θ_m schließen. Die durchschnittliche Abweichung der MCTS+θ_m zur MCTS und zum θ_m lässt aber nur einen qualitativen Rückschluss auf die Generalisierungsfähigkeit zu, da die bessere TLT der MCTS+θ_m nicht allein auf die Generalisierungsfähigkeit, sondern auch auf die generell bessere Lösungsqualität dieser Methode zurückzuführen ist. Das Ergebnis bestätigt, dass die starke Generalisierungsfähigkeit der MCTS+θ_m mit einem Trainingsframework noch weiter verbessert werden kann, sodass die Total Lead Time der MCTS+θ_m selbst für unbekannte Störungsszenarien fast das Niveau der PSO erreicht. Die Anforderung der Generalisierungsfähigkeit kann somit im Falle der MCTS+θ_m als erfüllt angesehen werden.

Implizit ist auch die Anforderung hinsichtlich einer **Reaktionszeit** von unter drei Minuten erfüllt (vgl. Abschnitt 3.3), da alle in der Validierung durchgeführten Versuchsläufe der MCTS+θ_m ein maximales Zeitlimit von drei Minuten aufweisen. Aufgrund der Validierungsergebnisse können alle Anforderungen an eine dynamische Terminierung einer flexiblen Fertigung für die MCTS+θ_m als betrachtet werden. Demzufolge lässt sich konstatieren, dass sich die in Kapitel 4 entwickelte Methode der MCTS+θ_m für die dynamische Terminierung einer flexiblen Fertigung eignet.

Die Validierungsergebnisse der Versuche im Open Shop lassen sich auch auf das Stanzerei-Environment übertragen (vgl. Abschnitt 5.3). Dort minimiert die MCTS+θ_m in allen Fällen die Total Lead Time deutlich besser als die MCTS oder das θ_m (vgl. Tabelle 5.6). Die TLT der PSO erreicht die MCTS+θ_m jedoch nicht. Andere Resultate zeigen sich, wenn unterschiedliche FAUF-Prioritäten eingeführt werden (vgl. Tabelle 5.7). In diesem Fall erreicht die MCTS+θ_m eine durchschnittlich bessere TLT als die PSO. Auch die Validierungsergebnisse aus dem OSSP bzgl. der Generalisierungsfähigkeit der MCTS+θ_m lassen sich auf das Stanzerei-Environment übertragen. So erreicht die MCTS+θ_m TLTs mit einer durchschnittlichen Abweichung von 0,845% gegenüber den TLTs der PSO, obwohl beim Training der MCTS+θ_m keine Störungen berücksichtigt werden (vgl. Tabelle 5.8). Die MCTS+θ_m bestätigt damit die starke Generalisierungsfähigkeit der Methode. Die durchschnittliche Abweichung der TLTs der MCTS+θ_m zu den Ergebnissen der PSO liegt im Open Shop mit 3,976% deutlich höher (vgl. Tabelle 5.5). Das lässt sich damit erklären, dass es im Open Shop wesentlich mehr mögliche Alternativen als im Stanzerei-Environment gibt. Aufgrund der großen Anzahl der Alternativen im Open Shop fällt die Generalisierungsfähigkeit im OSSP schlechter aus. Das OSSP ist allerdings ein Extremfall, der nicht einer Fertigung in der Realität entspricht. Mit einem Trainingsframework kann die Generalisierungsfähigkeit im OSSP stark verbessert werden. Für reale Anwendungen, wie die Stanzerei, kann aber auf das Trainingsframework verzichtet werden, da die Generalisierungsfähigkeit des MCTS+θ_m ausreicht, um alle Störungsszenarien gut lösen zu können (vgl. Tabelle 5.8).

Die im Abschnitt 4.2.5 entwickelte Methode des Clusterns kann erfolgreich dazu eingesetzt werden, die TLT der MCTS+θ_m weiter zu minimieren (vgl. Tabelle 5.9). Das Clustern der FAUFs verbessert die durchschnittliche Abweichung zur TLT der PSO von 1,611% bei der ungeclusterten MCTS+θ_m auf 0,108% für die geclusterte MCTS+θ_m (vgl. Tabellen 5.18 und 5.9). Die Methode des Clusterns hat somit die Fähigkeit zur Minimierung der Total Lead Time.

Es ist zu berücksichtigen, dass in der Validierung schwerpunktmäßig die Verbesserung der Lösungsqualität der in Kapitel 4 konzipierte Methode MCTS+θ_m, im Vergleich zu einer MCTS und einem θ_m untersucht wird. Die PSO wird in der Literatur als Benchmark-Metaheuristik verwendet, wenn es um die Lösungsqualität des OSSPs geht. Die Ergebnisse der PSO dienen als Vergleichswerte bzw. können als nahezu optimal angesehen werden. Die MCTS+θ_m erreicht in allen Validierungsszenarien bessere Ergebnisse als die MCTS oder das θ_m. Das zeigt, dass die in Kapitel 4 entwickelte Methode somit als sinnvolle Erweiterung einer MCTS angesehen werden kann. Weitere Untersuchungen bzgl. der Skalierbarkeit der Methode oder Benchmarks gegenüber anderen Methoden stehen aus. Zusätzlich können Teile der Methode gegen andere Methode ausgetauscht werden. So ist es denkbar, dass anstelle des θ_m auch ein sogenanntes ‚Graph Neural Network' eingesetzt werden kann [77].

6 Fazit und Ausblick

In diesem Kapitel werden die in der vorliegenden Arbeit entwickelte Methode und die Ergebnisse der Validierung zusammengefasst. Zusätzlich werden die in Abschnitt 1.3 definierten Ziele auf ihre Gültigkeit geprüft. Den Abschluss dieses Kapitels bildet ein Ausblick auf weitere, offene Forschungsansätze im Bereich der dynamischen Terminierung.

6.1 Fazit

Ein zunehmender Kostendruck und eine wachsende Anzahl an Mitbewerbern auf dem globalen Markt zwingen Unternehmen dazu, ihre Fertigung zu optimieren. Dabei spielen Indikatoren wie die Durchlaufzeit der Bauteile und die Maschinennutzung eine zentrale Rolle. Störungen, z B. Maschinenausfälle, führen dazu, dass die Fertigung nicht wie geplant ablaufen kann, und beeinflussen die genannten Indikatoren negativ. Eine Neuterminierung der Fertigung benötigt zu viel Zeit, sodass die Situation der Fertigung bei Fertigstellung der neuen Terminierung stark von der Ausgangssituation abweicht. Manuelle Eingriffe führen aufgrund der zahlreichen Zusammenhänge oft nicht zur optimalen Fertigungsauslastung. Angesichts dieser Problemstellung verfolgt die vorliegende Arbeit das Ziel, eine Methode für die dynamische Terminierung zu entwickeln, die die FAUFs im Störungsfall zeitnah umterminiert, sodass die Total Lead Time bestmöglich minimiert wird.

In Kapitel 3 werden aktuelle Ansätze, die genutzt werden können, um ein Terminierungs- bzw. Scheduling-Problem zu lösen, erklärt und beschrieben. Während analytische Methoden sowie Heuristiken bzw. Metaheuristiken in der Forschung umfangreich eingesetzt werden, um Scheduling-Probleme zu lösen, besteht im KI-Bereich und besonders im Fall von kombinierter KI-Methoden weiter Forschungsbedarf (vgl. Abschnitt 3.4). Zusätzlich werden Anforderungen an die dynamische Terminierung einer flexiblen Fertigung definiert (vgl. Abschnitt 3.3). In Kapitel 4 wird eine Methode zur dynamischen Terminierung entwickelt. Die Methode besteht aus einem Entscheidungsbaum, der MCTS, kombiniert mit einer Reinforcement-Learning-Methode, dem MADQN. Zusätzlich werden die Methoden ‚Action-Combination', ‚FAUF-Clustering' und ‚Trainingsframework' entwickelt, um die Anforderungen an die dynamische Terminierung zu erfüllen.

In der Validierung wird die in Kapitel 4 konzipierte Methode anhand verschiedener Szenarien auf ihre Anwendbarkeit überprüft (vgl. Kapitel 5). Dabei findet der erste Teil der Validierung im Open Shop - mit maximalen Freiheitsgraden bzw. maximaler Komplexität statt (vgl. Abschnitt 5.3.1). Dieser Teil der Validierung konzentriert sich darauf, einige grundsätzliche Überprüfung in Bezug auf die Erfüllung der vorher definierten Anforderungen vorzunehmen. Da das Open-Shop-Scheduling-Problem keiner realen Fertigung entspricht, wird im zweiten Teil der Validierung eine reale Fertigung, die Stanzerei, herangezogen. Die Validierungsergebnisse zeigen, dass die entwickelte Methode alle Anforderungen an eine dynamische Terminierung für die untersuchten Environments erfüllt.

Auf Grundlage der Validierungsergebnisse lassen sich die drei Ziele aus Abschnitt 1.3 bewerten. Das *erste Ziel* kann teilweise angenommen werden. Komplexe Optimierungsprobleme sind in der Regel nur schwierig optimal zu lösen. Bei Scheduling-Problemen gibt es Benchmarks, bei denen mit neuen Methoden und leistungsstärkerer Hardware zunehmend bessere Lösungen gefunden werden können. Aus diesem Grund wird in der Validierung die Metaheuristik PSO als Benchmarkmethode festgelegt. Sowohl in den Validierungsläufen im Open Shop als auch im Stanzerei-Environment findet die in dieser Arbeit entwickelte Methode bessere oder nahezu gleichwertige Lösungen, sodass das erste Ziel teilweise angenommen werden kann.

Das *zweite Ziel* ist teilweise erfüllt. Die entwickelte Methode löst das Open-Shop-Problem besser als die verglichene Heuristik und ist somit dazu in der Lage, auch komplexe Terminierungsprobleme gut zu lösen. Zusätzlich konnte nachgewiesen werden, dass die Streuung der Lösungsqualität gegenüber einer nicht unterstützten MCTS deutlich abnimmt, was einer Steigerung der Robustheit entspricht (vgl. Abbildung 5.6). Die Streuung konnte dabei nahezu um das 4-fache im Vergleich zur MCTS verringert werden. Trotzdem bleibt bei diesem Ansatz eine geringfügige Streuung vorhanden, da weder die MCTS noch das MADQN deterministisch arbeiten.

Das *dritte Ziel* kann umfänglich angenommen werden. Das in Abschnitt 4.2.6 entwickelte Trainingsframework ermöglicht eine deutliche Verbesserung der Generalisierungsfähigkeit im Open Shop (vgl. Tabelle 5.5). Werden beim Training Störungsszenarien betrachtet, erhöht sich die Trainingszeit aufgrund der Vergrößerung des Lösungsraums. Die Lösungsqualität im Störungsfall fällt in den Untersuchungen bei Verwendung des Trainingsframeworks deutlich besser aus als beim Training ohne Trainingsframework. Zusammen mit der Generalisierungsfähigkeit (vgl. Abbildung 5.8) der entwickelten Methode können so selbst Störungsszenarien, die nicht im Training berücksichtigt werden, nahezu optimal gelöst werden.

6.2 Ausblick

Die im Rahmen dieser Arbeit entwickelte Methode erfüllt die theoretischen Anforderungen an die dynamische Terminierung einer flexiblen Fertigung. Für den Einsatz in einer realen Fertigung müssen zusätzliche Randbedingungen bzgl. der Implementierung gegeben sein. In Abschnitt 3.1 wird die Funktionalität der dynamischen Terminierung der Ebene des Fertigungsmanagements zugeordnet. Für den realen Einsatz ist daher zu überlegen, wie eine IT-Systemarchitektur aussehen kann, die eine dynamische Terminierung zulässt. Neben Terminierungsinformationen aus dem APS-System werden Informationen über aktuelle Maschinenzustände und Status der einzelnen FAUFs benötigt.

Zusätzlich ist zu klären, in welcher Umgebung die entwickelte Methode ausgeführt werden soll. Grundsätzlich ist es denkbar, diese als Applikation in einer Cloud auszuführen. Dafür spricht die theoretisch unbegrenzte Rechenleistung der Cloud-Server, die eine schnelle Lösung des Optimierungsproblems ermöglicht. Aufgrund der Nähe zur Steuerungsebene (vgl. Abschnitt 3.1) ist aber auch eine Ausführung in einem bzw. mehreren Edge-Devices denkbar. Die Applikation läuft dabei lokal im Fertigungsnetz; so werden fertigungsrelevante Daten nicht auf Servern gespeichert, die mit dem Internet sind. Die Rechenleistung der Edge-Devices fällt im Vergleich zu leistungsstarken Workstations oder sogar Servern deutlich geringer aus; infolge dessen muss untersucht werden, ob und wie der Einsatz von Edge-Devices für die dynamische Terminierung sinnvoll ist [78].

Neben den möglichen Untersuchungen im Bereich der Implementierung der Methode können auch weitere Ansätze erforscht werden, um die entwickelte Methode zu verbessern. So könnten diese neuen Methoden das Potenzial besitzen, das vorliegende Optimierungsproblem mit weniger Rechenaufwand bzw. mit einer besseren Lösungsqualität zu lösen. Beispielsweise könnte an Stelle des verwendeten MADQNs auch ein sogenanntes ‚Graph Neural Network‘ (GNN) eingesetzt werden [77]. Das MADQN besteht aus mehreren einzelnen Agenten, die ihre Entscheidungen unabhängig voneinander treffen. Aus diesem Grund wird in Abschnitt 4.2.4 eine Kombination der Actions der einzelnen Agenten entwickelt. Bei einem GNN werden die Abhängigkeiten der Bauteile bzw. FAUFs in einen Graphen überführt. Beim Lösen des GNNs werden folglich alle Abhängigkeiten betrachtet, sodass keine Kombination verschiedener Actions notwendig ist. Zudem lassen erste Untersuchungen vermuten, dass die Generalisierungsfähigkeit eines GNNs besser als jene des MADQNs ausfallen kann.

Ein anderer Ansatz, das Optimierungsproblem zu lösen, kann der Einsatz von Quantencomputern sein bzw. der Einsatz des sogenannten ‚Quantum Annealing‘ [62] [143]. Quantencomputer zeichnen sich dadurch aus, dass sie Optimierungsprobleme schnell lösen können. Dafür muss das Optimierungsproblem jedoch entsprechend umformuliert werden.

A Anhang

A.1 Multi-Agent-Proximal-Policy-Optimization (MAPPO)

Die ‚Proximal-Policy-Optimization' ist eine Policy-Gradienten-Methode für einen Single-Agent-Ansatz [117]. Policy-Gradienten-Methoden parametrisieren die Policy (vgl. Abschnitt 2.4.4) direkt und ändern die Gewichte der Parametrisierung, um eine Policy zu erzeugen, die den Reward maximiert. Gradientenalgorithmen reagieren jedoch empfindlich auf die Anpassung der Schrittweite. Die Schrittweite bestimmt das Ausmaß, in dem die Parameter des neuronalen Netzes während des Trainings geändert werden; ist sie zu klein, wird das Training unnötig verlängert; ist sie zu groß, sind die Änderungen zu stark und führen nicht zur optimalen Policy. Durch die Verwendung der sogenannten ‚Clipped-Surrogate-Objective-Function' L^{Clip} kontrolliert die PPO die Änderung der Strategie während des Trainings[117]. Die PPO ermöglicht auf diese Weise feinfühlige Anpassungen der Schrittweite für jede Iteration und verhindert zusätzlich übermäßige Schwankungen der Parameter des neuronalen Netzes. [117]

Der MAPPO-Algorithmus basiert auf dem Single-Agenten-Ansatz mit PPO. Jedem Produkt wird bei der MAPPO-Methode ein Produkt-Agent zugewiesen (vgl. Abbildung A.1). Im Gegensatz zum Single-Agenten-Ansatz mit PPO teilen sich bei der MAPPO-Methode alle Produkt-Agenten einen zentralen Critic [118] [82]. Letzterer bewertet die Aktionen jedes einzelnen Produkt-Agenten im Kontext aller Produkt-Agenten. Zu diesem Zweck erhält der Critic den globalen Zustand und alle Aktionen des Zeitschritts t als Eingabe. Die L^{Clip}-Funktion der PPO berechnet den Parameter Θ, der angibt, wie stark die Parameter des neuronalen

Netzes des Produkt-Agenten angepasst werden müssen. Für die Anpassung des neuronalen Netzes des Critics werden neben den globalen States s_t auch die gewählten Actions a_t und die Rewards r_t benötigt. Die Produkt-Agenten maximieren demzufolge den Gesamt-Reward. Vergleichbar mit dem zentralisierten Critic des MADDPG [84] wird der Critic selbst nur zum Training benötigt. Während der Ausführung wird von jedem Produkt-Agenten eine Aktion basierend auf dem verfügbaren Zustand ausgegeben, die implizit die Aktionen der anderen Produkt-Agenten berücksichtigt.

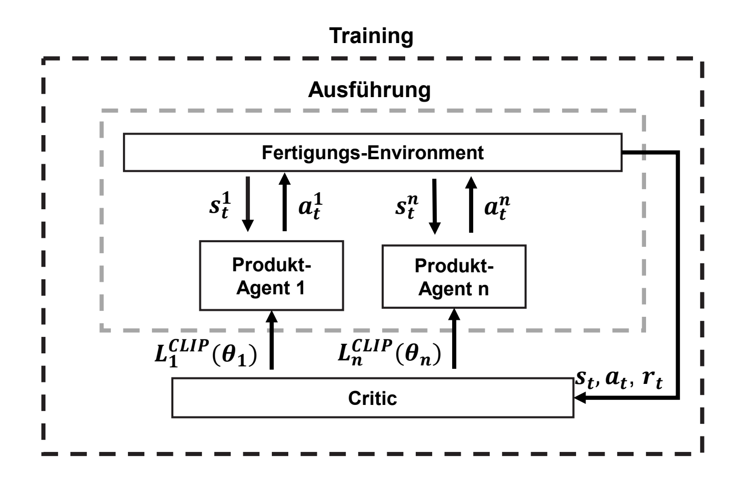


Abbildung A.1: Überblick über die MAPPO-Methode

Veröffentlichungen, die die MAPPO-Methode für die Fertigungsterminierung nutzen, haben gezeigt, dass diese Methode bestimmte Terminierungsprobleme erlernen und beherrschen kann. Es wurde aber auch gezeigt, dass die MAPPO-Methode nur funktioniert, wenn alle Produkte identisch sind bzw. die gleiche Priorität aufweisen. Besitzen Produkte unterschiedliche Prioritäten, kann der MAPPO kein Terminierungsergebnis erreichen, das einem globalen Optimum nahekommt. Haben alle Produkt-Agenten die gleiche Priorität, müssen diese nur Actions treffen, die ihre eigene Durchlaufzeit optimieren. Unterscheiden sich die Prioritäten ist es

notwendig, dass ein Produkt-Agent mit einer niedrigen Priorität auf einen Produkt-Agenten mit einer höheren Priorität wartet. Zusätzlich ist anzumerken, dass die Produkt-Agenten während der Ausführung nicht kommunizieren. Sie haben während des Trainings lediglich gewisse Ablaufreihenfolgen gelernt. Kommt es in der realen Fertigung zu einer Situation, die die Produkt-Agenten beim Training nicht durchlaufen haben, kann die MAPPO-Methode keine zufriedenstellende Lösung finden. Aufgrund dieser Eigenschaften eignet sich diese nicht für die dynamische Terminierung einer Fertigung. [82]

A.2 Arbeitspläne der Stanzerei

Für die Validierungsläufe im Stanzerei-Environment (vgl. Abschnitt 5.3) werden vier verschiedene Produkttypen mit unterschiedlichen Arbeitsplänen betrachtet. Jede der nachfolgenden Tabellen (vgl. Tabellen A.1 - A.5) bezieht sich auf einen der fünf Arbeitsvorgänge, die durchlaufen werden müssen. Die verschiedenen Produkttypen haben für den entsprechenden Arbeitsvorgang mindestens eine und maximal drei Maschinen zur Auswahl. Für die Abarbeitung des AVOs kann der Produkt-Agent lediglich eine Maschine wählen. Die Bearbeitungszeit des AVOs hängt von der jeweiligen Maschine ab und kann somit je AVO variieren.

Tabelle A.1: Arbeitsvorgang 10 im Stanzerei-Environment für vier unterschiedliche Produkttypen

	AVO 10					
	Maschinen-Nr.	**Bearbeitungszeit $[t]$**	**Maschinen-Nr.**	**Bearbeitungszeit $[t]$**	**Maschinen-Nr.**	**Bearbeitungszeit $[t]$**
Produkttyp 1	1	2	2	2	3	2
Produkttyp 2	1	2	2	2	3	2
Produkttyp 3	1	2	2	2	3	2
Produkttyp 4	1	2	2	2	3	2

Tabelle A.2: Arbeitsvorgang 20 im Stanzerei-Environment für vier unterschiedliche Produkttypen

	AVO 20					
	Maschinen-Nr.	**Bearbeitungszeit** $[t]$	**Maschinen-Nr.**	**Bearbeitungszeit** $[t]$	**Maschinen-Nr.**	**Bearbeitungszeit** $[t]$
Produkttyp 1	4	12	5	20		
Produkttyp 2	4	12	5	16		
Produkttyp 3	4	12				
Produkttyp 4	4	12				

Tabelle A.3: Arbeitsvorgang 30 im Stanzerei-Environment für vier unterschiedliche Produkttypen

	AVO 30					
	Maschinen-Nr.	**Bearbeitungszeit** $[t]$	**Maschinen-Nr.**	**Bearbeitungszeit** $[t]$	**Maschinen-Nr.**	**Bearbeitungszeit** $[t]$
Produkttyp 1	6	8	7	35		
Produkttyp 2	6	8	7	22		
Produkttyp 3	5	12	6	8		
Produkttyp 4	5	18	6	8		

Tabelle A.4: Arbeitsvorgang 40 im Stanzerei-Environment für vier unterschiedliche Produkttypen

	AVO 40					
	Maschinen-Nr.	**Bearbeitungszeit** $[t]$	**Maschinen-Nr.**	**Bearbeitungszeit** $[t]$	**Maschinen-Nr.**	**Bearbeitungszeit** $[t]$
Produkttyp 1	9	4				
Produkttyp 2	9	6				
Produkttyp 3	7	12				
Produkttyp 4	7	60				

Tabelle A.5: Arbeitsvorgang 50 im Stanzerei-Environment für vier unterschiedliche Produkttypen

	AVO 50					
	Maschinen-Nr.	**Bearbeitungszeit** $[t]$	**Maschinen-Nr.**	**Bearbeitungszeit** $[t]$	**Maschinen-Nr.**	**Bearbeitungszeit** $[t]$
Produkttyp 1	10	4	11	3	12	3
Produkttyp 2	10	3				
Produkttyp 3	10	5				
Produkttyp 4	8	59	11	3	12	3

Publikationsliste

Tabelle A.6: Publikationsliste

Jahr	Titel	Autoren	DOI	Konferenz
2020	Real Time Reaction Concept for Cyber Physical Production Systems	Oliver Lohse, Stefan Krause, Christopher Saal, Christian Lipp	10.1109/SIMS49386.2020.9121473	IEEE: 2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS)
2020	3D Model-Based Product Definition and Production – a Mind Change with Technical Hurdles	Christopher Saal, Christian Lipp, Oliver Lohse, Stefan Krause	10.1109/SIMS49386.2020.9121460	IEEE: 2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS)
2020	Manufacturing Operations Management for Smart Manufacturing – A Case Study	Michael Meyer-Hentschel, Oliver Lohse, Subba Rao, Raffaello Lepratti	10.1007/978-3-030-57993-7_11	Springer: 2020 Advances in Production Management Systems.
2021	Implementing an Online Scheduling Approach for Production with Multi Agent Proximal Policy Optimization (MAPPO)	Oliver Lohse, Noah Pütz, Korbinian Hörmann	10.1007/978-3-030-85914-5_62	Springer: 2021 Advances in Production Management Systems.
2021	Factory data management: Definition and differentiation from manufacturing operations management	Alexander Nowitschkow, Christopher Saal, Oliver Lohse	10.1109/ICIT46573.2021.9453563	IEEE: 2021 22nd IEEE International Conference on Industrial Technology (ICIT)
2022	Enhancing Monte-Carlo Tree Search with Multi-Agent Deep Q-Network in Open Shop Scheduling	Oliver Lohse, Aaron Haag, Tizian Dagner	10.1109/WCMEIM56910.2022.10021570	IEEE 2022 5th World Conference on Mechanical Engineering and Intelligent Manufacturing (WCMEIM)

Literaturverzeichnis

[1] abas Software GmbH ; abas Software GmbH (Hrsg.): *ABAS ERP: Ihr Vorsprung im Wettbewerb*. `https://abas-erp.com/sites/default/files/produktbroschuere-abas-erp-2021-web.pdf`. Aufgerufen am: 22.03.2022

[2] Acker, I. J.: Mehrkriterielles Job-Shop-Scheduling mit alternativen Maschinenfolgen. In: Wenger, W. (Hrsg.): *Business Excellence in Produktion und Logistik*. Wiesbaden : Gabler. – DOI 10.1007/978–3–8349–6688–9_4. – ISBN 978–3–8349–2700–2, S. 65–86

[3] Adam, D. : *Produktions-Management*. 9., überarb. Aufl., Nachdr. Wiesbaden : Gabler, 2001. – ISBN 9783409691178

[4] Agatonovic-Kustrin, S. ; Beresford, R. : Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. In: *Journal of Pharmaceutical and Biomedical Analysis* 22 (2000), Nr. 5, S. 717–727. – DOI 10.1016/S0731–7085(99)00272–1. – ISSN 07317085

[5] Aggarwal, C. C.: *Data Mining: The Textbook*. Aufl. 2015. Cham : Springer International Publishing, 2015. – ISBN 978–3–319–14141–1

[6] Aggarwal, C. C.: *Artificial Intelligence: A Textbook*. 1st edition 2021. Cham : Springer International Publishing, 2021. – ISBN 978–3–030–72357–6

[7] Amjad, M. K. ; Butt, S. I. ; Anjum, N. ; Chaudhry, I. A. ; Faping, Z. ; Khan, M. : A layered genetic algorithm with iterative diversification for

optimization of flexible job shop scheduling problems. In: *Advances in Production Engineering & Management* 15 (2020), Nr. 4, S. 377–389. – DOI 10.14743/apem2020.4.372. – ISSN 18546250

[8] Arnold, D. ; Isermann, H. ; Kuhn, A. ; Tempelmeier, H. ; Furmans, K. : *Handbuch Logistik*. 3., neu bearbeitete Auflage. Berlin and Ann Arbor : Springer and ProQuest eBook Central, 2008 (VDI). – ISBN 978–3–540–72928–0

[9] Artigues, C. ; Belmokhtar, S. ; Feillet, D. : A New Exact Solution Algorithm for the Job Shop Problem with Sequence-Dependent Setup Times. In: Kanade, T. (Hrsg.): *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* Bd. 3011. Berlin, Heidelberg : Springer Berlin Heidelberg. – DOI 10.1007/978–3–540–24664–0_3. – ISBN 978–3–540–21836–4, S. 37–49

[10] Aspen Technology, Inc. ; Aspen Technology, Inc. (Hrsg.): *Aspen Plant Scheduler ™ Family: Generate optimal production schedules to maximize profitability and meet service levels*. `https://www.aspentech.com/en/resources/brochure/aspen-plant-scheduler`. Aufgerufen am: 22.03.2022

[11] Babu MG, M. : *PP-DS with SAP S/4HANA*. 1. Auflage. New York, NY : Rheinwerk Publishing and SAP PRESS, 2020 (SAP PRESS Englisch). – ISBN 9781493218721

[12] Backhaus, K. ; Erichson, B. ; Plinke, W. ; Weiber, R. : *Multivariate Analysemethoden: Eine anwendungsorientierte Einführung*. 14., überarb. u. aktualisierte Aufl. 2016. Berlin, Heidelberg : Springer Berlin Heidelberg, 2016 (Springer eBook Collection). – ISBN 9783662460764

[13] Baer, S. ; Turner, D. ; Mohanty, P. K. ; Samsonov, V. ; Bakakeu, J. R. ; Meisen, T. : Multi Agent Deep Q-Network Approach for Online Job Shop Scheduling in Flexible Manufacturing. In: *International Conference on Manufacturing System and Multiple Machines* (2020), S. 1–9

[14] Barbu, A. ; Zhu, S.-C. : *Monte Carlo Methods*. 1st edition 2019. Puchong, Selangor D.E. : Springer Singapore and Springer, 2019. – ISBN 978–9811329708

[15] Barron, A. R.: Universal approximation bounds for superpositions of a sigmoidal function. In: *IEEE Transactions on Information Theory* 39 (1993), Nr. 3, S. 930–945. – DOI 10.1109/18.256500. – ISSN 0018–9448

[16] Benz, J. ; Höflinger, M. : *Logistikprozesse mit SAP: Eine anwendungsbezogene Einführung ; mit durchgehendem Fallbeispiel ; geeignet für SAP-Version 4.6A bis ECC 6.0 ; [mit Online-Service]*. 3., aktualisierte Aufl. Wiesbaden : Vieweg + Teubner, 2011 (Studium). – ISBN 978–3–8348–1484–5

[17] Blazewicz, J. ; Ecker, K. ; Pesch, E. ; Schmidt, G. ; Sterna, M. ; Weglarz, J. : *Handbook on scheduling: From theory to practice*. Second edition. Cham and Ann Arbor : Springer and ProQuest, 2019 (International handbooks on information systems). – ISBN 978–3–319–99849–7

[18] Brackel, T. : *Adaptive Steuerung flexibler Werkstattfertigungssysteme: Nutzung moderner Informations- und Kommunikationstechnologien zur effizienten Produktionssteuerung unter Echtzeitbedingungen*. Wiesbaden : Gabler Verlag / GWV Fachverlage GmbH Wiesbaden, 2009. – ISBN 978–3–8349–8066–3

[19] Braun, T. ; Görz, G. ; Schmid, U. : *Handbuch der Künstlichen Intelligenz*. 6. Auflage. Berlin and Boston : De Gruyter, 2021. – ISBN 9783110659948

[20] Breiman, L. ; Friedman, J. ; Charles, J. S. ; Olshen, R. : *Classification and Regression Trees*. London : Routledge, 1984 (Wadsworth statisticsprobability series). – ISBN 978–0412048418

[21] Browne, C. B. ; Powley, E. ; Whitehouse, D. ; Lucas, S. M. ; Cowling, P. I. ; Rohlfshagen, P. ; Tavener, S. ; Perez, D. ; Samothrakis, S. ; Colton, S. : A Survey of Monte Carlo Tree Search Methods. In: *IEEE Transactions on Computational Intelligence and AI in Games* 4 (2012), Nr. 1, S. 1–43. – DOI 10.1109/TCIAIG.2012.2186810. – ISSN 1943–068X

[22] Brucker, P. ; Jurisch, B. ; Sievers, B. : A branch and bound algorithm for the job-shop scheduling problem. In: *Discrete Applied Mathematics* 49 (1994), Nr. 1-3, S. 107–127. – DOI 10.1016/0166–218X(94)90204–6. – ISSN 0166218X

[23] Brucker, P. ; Knust, S. : *Complex scheduling*. 2. ed. Berlin : Springer, 2012 (GOR-Publications). – ISBN 978–3–642–23929–8

[24] Burke, E. K. ; Hyde, M. R. ; Kendall, G. ; Ochoa, G. ; Özcan, E. ; Woodward, J. R.: A Classification of Hyper-Heuristic Approaches: Revisited. In: Gendreau, M. (Hrsg.): *Handbook of metaheuristics* Bd. 272. Cham : Springer International Publishing. – DOI 10.1007/978–3–319–91086–4_14. – ISBN 978–3–319–91085–7, S. 453–477

[25] Cazenave, T. ; Jouandeau, N. : On the Parallelization of UCT. In: *Computer Games Workshop*. Amsterdam, Netherlands, Jun. 2007

[26] Chaslot, G. ; Bakkes, S. ; Szita, I. ; Spronck, P. : Monte-Carlo Tree Search: A New Framework for Game AI. In: *Artificial Intelligence and Interactive Digital Entertainment Conference* Bd. 4

[27] Chaslot, G. ; Saito, J.-T. ; Uiterwijk, J. W. ; Bouzy, B. ; van den Herik, H. J.: Monte-Carlo Strategies for Computer Go. In: *Proceedings of the 18th BeNeLux*, S. pp. 83–90

[28] Chaslot, G. M. J.-B. ; Winands, M. H. M. ; van Herik, H. J. ; Uiterwijk, J. W. H. M. ; Bouzy, B. : Progressive Strategies for Monte-Carlo Tree Search. In: *New Mathematics and Natural Computation* 04 (2008), Nr. 03, S. 343–357. – DOI 10.1142/S1793005708001094. – ISSN 1793–0057

[29] Chaslot, G. M. J.-B. ; Winands, M. H. ; van den Herik, H. J.: Parallel Monte-Carlo Tree Search. In: van den Herik, H. J. (Hrsg.): *Computers and Games* Bd. 5131. Berlin, Heidelberg : Springer Berlin Heidelberg, 2008. – ISBN 978–3–540–87608–3, S. 60–71

[30] Cobbe, K. ; Klimov, O. ; Hesse, C. ; Kim, T. ; Schulman, J. : *Quantifying Generalization in Reinforcement Learning*. `http://arxiv.org/pdf/1812.02341v3`. Aufgerufen am: 22.03.2022

[31] Coulom, R. : Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In: Hutchison, D. (Hrsg.): *Computers and Games* Bd. 4630. Berlin, Heidelberg : Springer Berlin Heidelberg. – DOI 10.1007/978–3–540–75538–8_7. – ISBN 978–3–540–75537–1, S. 72–83

[32] Cowling, P. ; Johansson, M. : Using real time information for effective dynamic scheduling. In: *European Journal of Operational Research* 139 (2002), Nr. 2, S. 230–244. – DOI 10.1016/S0377–2217(01)00355–1. – ISSN 03772217

[33] Dangelmaier, W. ; Mahajan, K. ; Seeger, T. ; Klopper, B. ; Aufenanger, M. : Simulation Assisted Optimization and Real-Time control Aspects of Flexible Production Systems Subject to Disturbances. In: *Proceedings - Winter Simulation Conference* 2007 (2007), S. 1785–1795. – DOI 10.1109/WSC.2006.322956

[34] Deroussi, L. ; Gourgand, M. ; Norre, S. : *New effective neighborhoods for the permutation flow shop problem*. `https://hal.archives-ouvertes.fr/hal-00678053`. Aufgerufen am: 22.03.2022

[35] Döbel, I. ; Dr. Hecker, D. ; Petersen, U. ; Rauschert, A. ; Schmitz, V. ; Dr. Voss, A. : *Zukunftsmarkt Künstliche Intelligenz: Potenziale und Anwendungen*. `https://www.bigdata-ai.fraunhofer.de/content/dam/bigdata/de/documents/Publikationen/KI-Potenzialanalyse_2017.pdf`. Aufgerufen am: 05.06.2021

[36] Durasević, M. ; Jakobović, D. : A survey of dispatching rules for the dynamic unrelated machines environment. In: *Expert Systems with Applications* 113 (2018), S. 555–569. – DOI 10.1016/j.eswa.2018.06.053. – ISSN 09574174

[37] Eberlin, S. ; Hock, B. : *Zuverlässigkeit und Verfügbarkeit technischer Systeme: Eine Einführung in die Praxis*. Wiesbaden : Springer Vieweg, 2014. – ISBN 978–3–658–03572–3

[38] Eversheim, W. : *Organisation in der Produktionstechnik Band 4: Fertigung und Montage*. 2. Berlin and Heidelberg : Springer, 1989 (VDI-Buch). – ISBN 978–3–642–61344–9

[39] Eversheim, W. : *Organisation in der Produktionstechnik: Band 1: Grundlagen*. Dritte, neu bearbeitete und erweiterte Auflage. Berlin and Heidelberg : Springer, 1996 (Studium und Praxis). – ISBN 978–3–642–87737–7

[40] Eversheim, W. : *Organisation in der Produktionstechnik 3: Arbeitsvorbereitung*. 4., bearbeitete und korrigierte Auflage. Berlin and Heidelberg : Springer, 2002 (VDI-Buch). – ISBN 978–3–642–56336–2

[41] Fischäder, H. : *Störungsmanagement in netzwerkförmigen Produktionssystemen*. Wiesbaden, Dt. Univ.-Verl, Zugl.: Ilmenau, Techn. Univ., Diss., 2005, 2007

[42] Garey, M. R. ; Johnson, D. S. ; Sethi, R. : The Complexity of Flowshop and Jobshop Scheduling. In: *Mathematics of Operations Research* 1 (1976), Nr. 2, S. 117–129. – DOI 10.1287/moor.1.2.117. – ISSN 0364–765X

[43] Gelly, S. ; Silver, D. : Monte-Carlo tree search and rapid action value estimation in computer Go. In: *Artificial Intelligence* 175 (2011), Nr. 11, S. 1856–1875. – DOI 10.1016/j.artint.2011.03.007. – ISSN 00043702

[44] Gelly, S. ; Wang, Y. : Exploration exploitation in Go: UCT for Monte-Carlo Go. In: *NIPS: Neural Information Processing Systems Conference On-line trading of Exploration and Exploitation Workshop*

[45] Georgila, K. ; Nelson, C. ; Traum, D. : Single-Agent vs. Multi-Agent Techniques for Concurrent Reinforcement Learning of Negotiation Dialogue Policies. In: Toutanova, K. (Hrsg.) ; Wu, H. (Hrsg.): *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume*

1: Long Papers). Stroudsburg, PA, USA : Association for Computational Linguistics, S. 500–510

[46] Géron, A. : *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. Second edition. Beijing and Boston and Farnham and Sebastopol and Tokyo : O'Reilly, 2019. – ISBN 9781492032649

[47] Goldberg, D. E. ; Holland, J. H.: Genetic Algorithms and Machine Learning. In: *Machine Learning* 3 (1988), Nr. 2/3, S. 95–99. – DOI 10.1023/A:1022602019183. – ISSN 0885–6125

[48] Gonzalez, T. ; Sahni, S. : Open Shop Scheduling to Minimize Finish Time. In: *Journal of the ACM* 23 (1976), Nr. 4, S. 665–679. – DOI 10.1145/321978.321985. – ISSN 0004–5411

[49] González Vila, T. (Hrsg.) ; Tucker, A. B. (Hrsg.): *Computer science and software engineering*. 3. ed. Boca Raton, Fla. : CRC Press, 2014 (Computing handbook). – ISBN 9781439898529

[50] Greschke, P. : *Matrix-Produktion: Konzept einer taktunabhängigen Fließfertigung*. 1. Auflage. Norderstedt : BoD – Books on Demand, 2020. – ISBN 3752670002

[51] Gruat-La-Forme, F. A. ; Botta-Genoulaz, V. ; Campagne, J.-P. ; Millet, P.-A. : Advanced Planning and Scheduling system: An overview of gaps and potential sample solutions. In: *International Conference on Industrial Engineering and Systems Management*. Marrakech, Morocco, Mai 2005, 683-695. – 13 pages

[52] Guillaume Maurice Jean-Bernard Chaslot: *Monte-Carlo Tree Search*. Maastricht, Universität Maastricht, Dissertation

[53] Günther, H.-O. ; Tempelmeier, H. : *Produktion und Logistik: Supply Chain und Operations Management*. 11., verb. Aufl. Norderstedt : BoD, 2014. – ISBN 978–3735721952

[54] Hatiboglu, B. ; Schuler, S. ; Bildstein, A. ; Hämmerle, M. : *Einsatzfelder von künstlicher Intelligenz im Produktionsumfeld: Kurzstudie im Rahmen von „100 Orte für Industrie 4.0 in Baden-Württemberg“ März 2019*. http://publica.fraunhofer.de/eprints/urn_nbn_de_0011-n-5491073.pdf. Aufgerufen am: 22.03.2022

[55] Herroelen, W. ; Leus, R. : Robust and reactive project scheduling: a review and classification of procedures. In: *International Journal of Production Research* 42 (2004), Nr. 8, S. 1599–1620. – DOI 10.1080/00207540310001638055. – ISSN 0020–7543

[56] Hester, T. ; Vecerik, M. ; Pietquin, O. ; Lanctot, M. ; Schaul, T. ; Piot, B. ; Horgan, D. ; Quan, J. ; Sendonaris, A. ; Dulac-Arnold, G. ; Osband, I. ; Agapiou, J. ; Leibo, J. Z. ; Gruslys, A. : *Deep Q-learning from Demonstrations*. http://arxiv.org/pdf/1704.03732v4. Aufgerufen am: 22.03.2022

[57] Höck, M. : *Betriebswirtschaftliche Forschung Zur Unternehmensführung Ser*. Bd. v.33: *Produktionsplanung und -Steuerung Einer Flexiblen Fertigung: Ein Prozeßorientierter Ansatz*. Wiesbaden : Springer Gabler. in Springer Fachmedien Wiesbaden GmbH, 1998. – ISBN 9783663111597

[58] Hofmann, C. ; Krahe, C. ; Stricker, N. ; Lanza, G. : Autonomous production control for matrix production based on deep Q-learning. In: *Procedia CIRP* 88 (2020), S. 25–30. – DOI 10.1016/j.procir.2020.05.005. – ISSN 22128271

[59] Holland, J. H.: Genetic Algorithms. In: *Scientific American* 267 (1992), Nr. 1, S. 66–72. – DOI 10.1038/scientificamerican0792–66. – ISSN 0036–8733

[60] Hornik, K. ; Stinchcombe, M. ; White, H. : Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. In: *Neural Networks* 3 (1990), Nr. 5, S. 551–560. – DOI 10.1016/0893–6080(90)90005–6. – ISSN 08936080

[61] Hu, J. ; Wellman, M. P.: Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm. In: *Proceedings of the Fifteenth International*

Conference on Machine Learning. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc, 1998 (ICML ’98). – ISBN 1558605568, S. 242–250

[62] Ikeda, K. ; Nakamura, Y. ; Humble, T. S.: Application of Quantum Annealing to Nurse Scheduling Problem. In: *Scientific reports* 9 (2019), Nr. 1, S. 12837. – DOI 10.1038/s41598–019–49172–3

[63] International Electrotechnical Commission: *Enterprise-control system integration: Part 1: Models and terminology*. 1. Geneva, 2013

[64] Jennings, N. R. ; Sycara, K. ; Wooldridge, M. : A Roadmap of Agent Research and Development. In: *Autonomous Agents and Multi-Agent Systems* 1 (1998), Nr. 1, S. 7–38. – DOI 10.1023/A:1010090405266. – ISSN 13872532

[65] Jo, T. : *Machine Learning Foundations: Supervised, unsupervised, and advanced learning*. S.l. : Springer, 2021. – ISBN 978–3–030–65899–1

[66] John McCarthy ; Stanford University (Hrsg.): *What is artificial intelligence?* `http://www-formal.stanford.edu/jmc/whatisai.pdf`. Aufgerufen am: 12.08.2021

[67] Katehakis, M. N. ; Veinott, A. F.: The Multi-Armed Bandit Problem: Decomposition and Computation. In: *Mathematics of Operations Research* 12 (1987), Nr. 2, S. 262–268. – DOI 10.1287/moor.12.2.262. – ISSN 0364–765X

[68] Kennedy, J. ; Eberhart, R. : Particle swarm optimization. In: *Proceedings of ICNN’95 - International Conference on Neural Networks*, IEEE, 1995. – ISBN 0–7803–2768–3, S. 1942–1948

[69] Kirk, R. ; Zhang, A. ; Grefenstette, E. ; Rocktäschel, T. : *A Survey of Generalisation in Deep Reinforcement Learning*. `http://arxiv.org/pdf/2111.09794v5`. Aufgerufen am: 22.03.2022

[70] Kocsis, L. ; Szepesvári, C. : Bandit Based Monte-Carlo Planning. In: Hutchison, D. (Hrsg.): *Machine Learning: ECML 2006* Bd. 4212. Berlin, Heidelberg : Springer Berlin Heidelberg. – DOI 10.1007/11871842_29. – ISBN 978–3–540–45375–8, S. 282–293

[71] Ku, W.-Y. ; Beck, J. C.: Mixed Integer Programming models for job shop scheduling: A computational analysis. In: *Computers & Operations Research* 73 (2016), S. 165–173. – DOI 10.1016/j.cor.2016.04.006. – ISSN 03050548

[72] Kubiak, W. : *Springer eBook Collection*. Bd. 325: *A Book of Open Shop Scheduling: Algorithms, Complexity and Applications*. 1st ed. 2022. Cham : Springer International Publishing and Imprint Springer, 2022. – ISBN 9783030910259

[73] Kurbel, K. : *Enterprise Resource Planning und Supply Chain Management in der Industrie: Von MRP bis Industrie 4.0*. 8., vollst. überarb. und erw. Auflage. Berlin and Boston : De Gruyter Oldenbourg, 2016 (De Gruyter Studium). – ISBN 978–3110441680

[74] Laterre, A. ; Fu, Y. ; Jabri, M. K. ; Cohen, A.-S. ; Kas, D. ; Hajjar, K. ; Dahl, T. S. ; Kerkeni, A. ; Beguir, K. : *Ranked Reward: Enabling Self-Play Reinforcement Learning for Combinatorial Optimization*. `http://arxiv.org/pdf/1807.01672v3`. Aufgerufen am: 22.03.2022

[75] Lenstra, J. K. ; Rinnooy Kan, A. : Computational Complexity of Discrete Optimization Problems. In: *Discrete Optimization I, Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium* Bd. 4. Elsevier. – DOI 10.1016/s0167–5060(08)70821–5. – ISBN 9780444853226, S. 121–140

[76] Lerch, F. ; Ultsch, A. ; Lötsch, J. : Distribution Optimization: An evolutionary algorithm to separate Gaussian mixtures. In: *Scientific reports* 10 (2020), Nr. 1, S. 648. – DOI 10.1038/s41598–020–57432–w

[77] Li, J. ; Dong, X. ; Zhang, K. ; Han, S. : Solving Open Shop Scheduling Problem via Graph Attention Neural Network. In: *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE, 2020. – ISBN 978–1–7281–9228–4, S. 277–284

[78] Lin, C.-C. ; Deng, D.-J. ; Chih, Y.-L. ; Chiu, H.-T. : Smart Manufacturing Scheduling With Edge Computing Using Multiclass Deep Q Network. In: *IEEE Transactions on Industrial Informatics* 15 (2019), Nr. 7, S. 4276–4284. – DOI 10.1109/TII.2019.2908210. – ISSN 1551–3203

[79] Liu, A. ; Liang, Y. ; Liu, J. ; van den Broeck, G. ; Chen, J. : *On Effective Parallelization of Monte Carlo Tree Search*. http://arxiv.org/pdf/2006.08785v2. Aufgerufen am: 22.03.2022

[80] Liu, C.-L. ; Chang, C.-C. ; Tseng, C.-J. : Actor-Critic Deep Reinforcement Learning for Solving Job Shop Scheduling Problems. In: *IEEE Access* 8 (2020), S. 71752–71762. – DOI 10.1109/ACCESS.2020.2987820

[81] Lohse, O. ; Krause, S. ; Saal, C. ; Lipp, C. : Real Time Reaction Concept for Cyber Physical Production Systems. In: *2020 3rd International Symposium on Small-scale Intelligent Manufacturing Systems (SIMS)*, IEEE, 62020. – ISBN 978–1–7281–6419–9, S. 1–5

[82] Lohse, O. ; Pütz, N. ; Hörmann, K. : Implementing an Online Scheduling Approach for Production with Multi Agent Proximal Policy Optimization (MAPPO). In: Dolgui, A. (Hrsg.): *Advances in Production Management Systems. Artificial Intelligence for Sustainable and Resilient Production Systems* Bd. 634. Cham : Springer International Publishing. – DOI 10.1007/978–3–030–85914–5_62. – ISBN 978–3–030–85913–8, S. 586–595

[83] Lorenz, U. : *Reinforcement learning: Aktuelle Ansätze verstehen - mit Beispielen in Java und Greenfoot*. Berlin : Springer Vieweg, 2020. – ISBN 3662616505

[84] Lowe, R. ; Wu, Y. ; Tamar, A. ; Harb, J. ; Abbeel, P. ; Mordatch, I. : *Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments*. http://arxiv.org/pdf/1706.02275v4. Aufgerufen am: 22.03.2022

[85] M. Bagirov, A. : *Partitional Clustering Via Nonsmooth Optimization: Clustering Via Optimization*. Cham : Springer International Publishing AG, 2020 (Unsupervised and Semi-Supervised Learning). – ISBN 9783030378264

[86] Mahmoodi, F. ; Dooley, K. J. ; Starr, P. J.: An investigation of dynamic group scheduling heuristics in a job shop manufacturing cell. In: *International Journal of Production Research* 28 (1990), Nr. 9, S. 1695–1711. – DOI 10.1080/00207549008942824. – ISSN 0020–7543

[87] Majid Abdolrazzagh-Nezhad ; Salwani Abdullah: Job Shop Scheduling: Classification, Constraints and Objective Functions. In: *World Academy of Science, Engineering and Technology, International Journal of Computer and Information Engineering* 11 (2017), S. 429–434

[88] Majumder, A. : *Deep Reinforcement Learning in Unity*. Berkeley, CA : Apress, 2021. – ISBN 978–1–4842–6502–4

[89] Martin, R. C. ; Feathers, M. C.: *Clean code: A handbook of agile software craftsmanship*. Upper Saddle River NJ : Prentice Hall, 2009 (Robert C. Martin series). – ISBN 978–0–13–235088–4

[90] Mauergauz, Y. : *Advanced Planning and Scheduling in Manufacturing and Supply Chains*. 1st ed. 2016. Cham : Springer International Publishing and Springer, 2016. – ISBN 978–3–319–27521–5

[91] Mladenović, N. ; Hansen, P. : Variable neighborhood search. In: *Computers & Operations Research* 24 (1997), Nr. 11, S. 1097–1100. – DOI 10.1016/S0305–0548(97)00031–2. – ISSN 03050548

[92] Mnih, V. ; Kavukcuoglu, K. ; Silver, D. ; Graves, A. ; Antonoglou, I. ; Wierstra, D. ; Riedmiller, M. : *Playing Atari with Deep Reinforcement Learning*. `http://arxiv.org/pdf/1312.5602v1`. Aufgerufen am: 22.03.2022

[93] Mnih, V. ; Kavukcuoglu, K. ; Silver, D. ; Rusu, A. A. ; Veness, J. ; Bellemare, M. G. ; Graves, A. ; Riedmiller, M. ; Fidjeland, A. K. ; Ostrovski, G. ; Petersen, S. ; Beattie, C. ; Sadik, A. ; Antonoglou, I. ; King, H. ; Kumaran, D. ; Wierstra, D. ; Legg, S. ; Hassabis, D. : Human-level control through deep reinforcement learning. In: *Nature* 518 (2015), Nr. 7540, S. 529–533. – DOI 10.1038/nature14236

[94] Moshkov, M. : *Intelligent Systems Reference Library*. Bd. v.179: *Comparative Analysis of Deterministic and Nondeterministic Decision Trees*. Cham : Springer International Publishing AG, 2020. – ISBN 9783030417284

[95] Nguyen, S. ; Zhang, M. ; Johnston, M. ; Tan, K. C.: Genetic Programming for Job Shop Scheduling. In: Bansal, J. C. (Hrsg.) ; Singh, P. K. (Hrsg.) ; Pal, N. R. (Hrsg.): *Evolutionary and Swarm Intelligence Algorithms* Bd. 779. Cham : Springer International Publishing. – DOI 10.1007/978–3–319–91341–4_8. – ISBN 978–3–319–91339–1, S. 143–167

[96] Noback, M. : *Principles of Package Design: Creating Reusable Software Components*. 1st edition. New York : Apress, 2018. – ISBN 978–1–4842–4118–9

[97] Nwankpa, C. ; Ijomah, W. ; Gachagan, A. ; Marshall, S. : *Activation Functions: Comparison of trends in Practice and Research for Deep Learning*. `http://arxiv.org/pdf/1811.03378v1`. Aufgerufen am: 22.03.2022

[98] Olson, D. L. (Hrsg.) ; Lauhoff, G. (Hrsg.): *Descriptive data mining*. Second edition. Singapore : Springer, 2019 (Computational Risk Management). – ISBN 978–981–13–7180–6

[99] Oracle ; Oracle (Hrsg.): *JD Edwards EnterpriseOne Applications: Shop Floor Management Implementation Guide: E15143-11, Release 9.1*. `https://docs.oracle.com/cd/E16582_01/doc.91/e15143/toc.htm`. Aufgerufen am: 22.03.2022

[100] Osman, I. H.: Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. In: *Annals of Operations Research* 41 (1993), Nr. 4, S. 421–451. – DOI 10.1007/BF02023004. – ISSN 0254–5330

[101] Ouelhadj, D. ; Petrovic, S. : A survey of dynamic scheduling in manufacturing systems. In: *Journal of Scheduling* 12 (2009), Nr. 4, S. 417–431. – DOI 10.1007/s10951–008–0090–8. – ISSN 1094–6136

[102] Paaß, G. ; Hecker, D. : *Künstliche Intelligenz: Was steckt hinter der Technologie der Zukunft?* 1. Auflage 2020. Wiesbaden : Springer Fachmedien Wiesbaden, 2020 (Springer eBook Collection). – ISBN 9783658302115

[103] Panzer, M. ; Bender, B. ; Gronau, N. : Deep Reinforcement Learning In Production Planning And Control: A Systematic Literature Review. . – DOI 10.15488/11238

[104] Pezzella, F. ; Morganti, G. ; Ciaschetti, G. : A genetic algorithm for the Flexible Job-shop Scheduling Problem. In: *Computers & Operations Research* 35 (2008), Nr. 10, S. 3202–3212. – DOI 10.1016/j.cor.2007.02.014. – ISSN 03050548

[105] Pham, D. T. ; Afify, A. A.: Machine-learning techniques and their applications in manufacturing. In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 219 (2005), Nr. 5, S. 395–412. – DOI 10.1243/095440505X32274. – ISSN 0954–4054

[106] Pinedo, M. : *Scheduling: Theory, algorithms, and systems*. Fifth edition. Cham : Springer, 2016. – ISBN 978–3–319–26578–0

[107] Rahmani Hosseinabadi, A. A. ; Vahidi, J. ; Saemi, B. ; Sangaiah, A. K. ; Elhoseny, M. : Extended Genetic Algorithm for solving open-shop scheduling problem. In: *Soft Computing* 23 (2019), Nr. 13, S. 5099–5116. – DOI 10.1007/s00500–018–3177–y. – ISSN 1432–7643

[108] REFA - Verband für Arbeitsstudien und Betriebsorganisation: *Methodenlehre der Betriebsorganisation: Planung und Steuerung - Teil 2*. 1. Aufl. München : Hanser, 1991. – ISBN 978–3446163508

[109] REFA - Verband für Arbeitsstudien und Betriebsorganisation: *Datenermittlung*. München : Hanser, 1997 (Methodenlehre der Betriebsorganisation). – ISBN 9783446190597

[110] Rinciog, A. ; Mieth, C. ; Scheikl, P. M. ; Meyer, A. : Sheet-Metal Production Scheduling Using AlphaGo Zero. In: *Conference on Production Systems and Logistics*. – DOI 10.15488/9676, S. 342–352

[111] Rosin, C. D.: Multi-armed bandits with episode context. In: *Annals of Mathematics and Artificial Intelligence* 61 (2011), Nr. 3, S. 203–230. – DOI 10.1007/s10472–011–9258–6. – ISSN 1012–2443

[112] Rousseeuw, P. J.: Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. In: *Journal of Computational and Applied Mathematics* 20 (1987), S. 53–65. – DOI 10.1016/0377–0427(87)90125–7. – ISSN 03770427

[113] Russell, S. : *Pearson Studium / Informatik*. Bd. 4098: *Künstliche Intelligenz: Ein moderner Ansatz*. 3., aktualisierte Aufl. München : Pearson Studium, 2012. – ISBN 9783868940985

[114] Sastry, K. ; Goldberg, D. ; Kendall, G. : Genetic Algorithms. In: Burke, E. K. (Hrsg.) ; Kendall, G. (Hrsg.): *Search Methodologies*. Boston, MA : Springer US. – DOI 10.1007/0–387–28356–0_4. – ISBN 978–0–387–23460–1, S. 97–125

[115] Schneeweiß, C. ; Söhner, V. : *Schriften zur Quantitativen Betriebswirtschaftslehre*. Bd. 2: *Kapazitätsplanung bei moderner Fließfertigung*. Heidelberg : Physica-Verlag HD, 1991. – ISBN 978–3–662–12136–8

[116] Schuh, G. ; Stich, V. : *Produktionsplanung und -steuerung 1: Grundlagen der PPS*. 4. Aufl. 2012. Berlin Heidelberg : Springer Berlin Heidelberg, 2012 (VDI-Buch). – ISBN 978–3–642–25423–9

[117] Schulman, J. ; Wolski, F. ; Dhariwal, P. ; Radford, A. ; Klimov, O. : *Proximal Policy Optimization Algorithms*. `http://arxiv.org/pdf/1707.06347v2`. Aufgerufen am: 22.03.2022

[118] Si, J. ; Barto, A. G. ; Powell, W. B. ; Wunsch, D. : *Handbook of Learning and Approximate Dynamic Programming*. Wiley-IEEE Press, 2004. – 359–380 S. – ISBN 9780470544785

[119] Siemens Digital Industries Software ; Siemens Digital Industries Software (Hrsg.): *Opcenter Scheduling: Advanced scheduling software*. `https://siemens.mindsphere.io/content/dam/cloudcraze-mind`

sphere-assets/plm/opcenter-scheduling-and-cloud-connected-viewer/Siemens%20SW%20Opcenter%20scheduling%20Flyer.pdf. Aufgerufen am: 22.03.2022

[120] Silver, D. ; Huang, A. ; Maddison, C. J. ; Guez, A. ; Sifre, L. ; van den Driessche, G. ; Schrittwieser, J. ; Antonoglou, I. ; Panneershelvam, V. ; Lanctot, M. ; Dieleman, S. ; Grewe, D. ; Nham, J. ; Kalchbrenner, N. ; Sutskever, I. ; Lillicrap, T. ; Leach, M. ; Kavukcuoglu, K. ; Graepel, T. ; Hassabis, D. : Mastering the game of Go with deep neural networks and tree search. In: *Nature* 529 (2016), Nr. 7587, S. 484–489. – DOI 10.1038/nature16961

[121] Silver, D. ; Hubert, T. ; Schrittwieser, J. ; Antonoglou, I. ; Lai, M. ; Guez, A. ; Lanctot, M. ; Sifre, L. ; Kumaran, D. ; Graepel, T. ; Lillicrap, T. ; Simonyan, K. ; Hassabis, D. : *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.* http://arxiv.org/pdf/1712.01815v1. Aufgerufen am: 22.03.2022

[122] Specht, D. (Hrsg.): *Weiterentwicklung der Produktion: Tagungsband der Herbsttagung 2008 der Wissenschaftlichen Kommission Produktionswirtschaft im VHB.* 1. Aufl. Wiesbaden : Gabler, 2009 (Gabler Research : Beiträge zur Produktionswirtschaft). – ISBN 978–3–8349–1830–7

[123] Stadtler, H. (Hrsg.): *Supply chain management and advanced planning: Concepts, models, software, and case studies*. 5. ed. Heidelberg and New York, NY and Dordrecht and London and Berlin : Springer, 2015 (Springer texts in business and economics). – ISBN 978–3–642–55308–0

[124] Stanković, A. ; Petrović, G. ; Ćojbašić, Ž. ; Marković, D. : An application of metaheuristic optimization algorithms for solving the flexible job-shop scheduling problem. In: *Operational Research in Engineering Sciences: Theory and Applications* 3 (2020), Nr. 3, S. 13–28. – DOI 10.31181/oresta20303013s. – ISSN 26201607

[125] Stoop, P. P. ; Wiers, V. C.: The complexity of scheduling in practice. In: *International Journal of Operations & Production Management* 16 (1996), Nr. 10, S. 37–53. – DOI 10.1108/01443579610130682. – ISSN 0144–3577

[126] Sun, Y. ; Zhang, C. ; Gao, L. ; Wang, X. : Multi-objective optimization algorithms for flow shop scheduling problem: a review and prospects. In: *The International Journal of Advanced Manufacturing Technology* 55 (2011), Nr. 5-8, S. 723–739. – DOI 10.1007/s00170–010–3094–4. – ISSN 0268–3768

[127] Sutton, R. S. ; Barto, A. : *Reinforcement learning: An introduction*. Second edition. Cambridge, MA and London : The MIT Press, 2018 (Adaptive computation and machine learning). – ISBN 9780262039246

[128] Tassel, P. ; Gebser, M. ; Schekotihin, K. : *A Reinforcement Learning Environment For Job-Shop Scheduling*. `http://arxiv.org/pdf/2104.03760v1`. Aufgerufen am: 22.03.2022

[129] Thommen, J.-P. ; Achleitner, A.-K. ; Gilbert, D. U. ; Hachmeister, D. ; Jarchow, S. ; Kaiser, G. : *Allgemeine Betriebswirtschaftslehre: Umfassende Einführung aus managementorientierter Sicht*. 9., vollständig überarbeitete Auflage. Wiesbaden : Springer Gabler, 2020. – ISBN 978–3658272456

[130] Turgut, Y. ; Bozdag, C. E.: Deep Q-Network Model for Dynamic Job Shop Scheduling Problem Based on Discrete Event Simulation. In: *2020 Winter Simulation Conference (WSC)*, IEEE, 2020. – ISBN 978–1–7281–9499–8, S. 1551–1559

[131] Turing, A. M.: I.—Computing Machinery And Intelligence. In: *Mind* LIX (1950), Nr. 236, S. 433–460. – DOI 10.1093/mind/LIX.236.433. – ISSN 0026–4423

[132] van Hoorn, J. J.: The Current state of bounds on benchmark instances of the job-shop scheduling problem. In: *Journal of Scheduling* 21 (2018), Nr. 1, S. 127–128. – DOI 10.1007/s10951–017–0547–8. – ISSN 1094–6136

[133] Wan, X. ; Evers, P. T. ; Dresner, M. E.: Too much of a good thing: The impact of product variety on operations and sales performance. In: *Journal of Operations Management* 30 (2012), Nr. 4, S. 316–324. – DOI 10.1016/j.jom.2011.12.002. – ISSN 02726963

[134] Wannenwetsch, H. : *Integrierte Materialwirtschaft und Logistik: Beschaffung, Logistik, Materialwirtschaft und Produktion*. Berlin Heidelberg : Springer-Verlag Berlin Heidelberg, 2010. – ISBN 978–3–540–89773–6

[135] Watermeyer, K. : *Ablaufplanung mit alternativen Prozessplänen*. Wiesbaden : Springer Gabler, 2016. – ISBN 9783658120924

[136] Watkins, C. J. C. H. ; Dayan, P. : Q-learning. In: *Machine Learning* 8 (1992), Nr. 3-4, S. 279–292. – DOI 10.1007/BF00992698. – ISSN 0885–6125

[137] Wenger, W. ; Geiger, M. J. ; Kleine, A. : *Business Excellence in Produktion und Logistik: Festschrift für Prof. Dr. Walter Habenicht*. 1. Aufl. Gabler Verlag / Springer Fachmedien Wiesbaden GmbH Wiesbaden. – ISBN 978–3–8349–2700–2

[138] Wierzchoń, S. T. ; Kłopotek, M. : *Studies in Big Data*. Bd. 34: *Modern Algorithms of Cluster Analysis*. 1st edition 2018. Cham : Springer International Publishing, 2018. – ISBN 9783319693071

[139] Winder, P. : *Reinforcement Learning: Industrial Applications of Intelligent Agents 1st Edition*. Farnham : O'Reilly UK Ltd., 2020. – ISBN 978–1098114831

[140] Xhafa, F. (Hrsg.): *Studies in computational intelligence*. Bd. Vol. 128: *Metaheuristics for scheduling in industrial and manufacturing applications*. Berlin and Heidelberg : Springer, 2008. – ISBN 978–3–540–78984–0

[141] Yeh, C.-C. ; Lu, H.-L. ; Yeh, J.-J. ; Huang, S.-K. : Path Exploration Based on Monte Carlo Tree Search for Symbolic Execution. In: *2017 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, IEEE, 2017. – ISBN 978–1–5386–4203–0, S. 33–37

[142] Zhou, L. ; Zhang, L. ; Horn, B. K.: Deep reinforcement learning-based dynamic scheduling in smart manufacturing. In: *Procedia CIRP* 93 (2020), S. 383–388. – DOI 10.1016/j.procir.2020.05.163. – ISSN 22128271

[143] Zielewski, M. R. ; Agung, M. ; Egawa, R. ; Takizawa, H. : Improving Quantum Annealing Performance on Embedded Problems. In: *Supercomputing Frontiers and Innovations* 7 (2020), Nr. 4, S. 32–48. – DOI 10.14529/jsfi200403. – ISSN 23138734

[144] Zurada, J. M.: *Introduction to artificial neural systems*. St. Paul : West, 1994. – ISBN 0314933913

REIHE INFORMATIONSMANAGEMENT IM ENGINEERING KARLSRUHE (ISSN 1860-5990)

Band 1 – 2005
SEIDEL, MICHAEL
Methodische Produktplanung. Grundlagen, Systematik und Anwendung im Produktentstehungsprozess.
ISBN 3-937300-51-1

Band 1 – 2006
PRIEUR, MICHAEL
Functional elements and engineering template-based product development process. Application for the support of stamping tool design.
ISBN 3-86644-033-2

Band 2 – 2006
GEIS, STEFAN RAFAEL
Integrated methodology for production related risk management of vehicle electronics (IMPROVE).
ISBN 3-86644-011-1

Band 1 – 2007
GLOSSNER, MARKUS
Integrierte Planungsmethodik für die Presswerkneutypplanung in der Automobilindustrie.
ISBN 978-3-86644-179-8

Band 2 – 2007
MAYER-BACHMANN, ROLAND
Integratives Anforderungsmanagement. Konzept und Anforderungsmodell am Beispiel der Fahrzeugentwicklung.
ISBN 978-3-86644-194-1

Band 1 – 2008
MBANG SAMA, ACHILLE
Holistic integration of product, process and resources integration in the automotive industry using the example of car body design and production. Product design, process modeling, IT implementation and potential benefits.
ISBN 978-3-86644-243-6

Band 2 – 2008
WEIGT, MARKUS
Systemtechnische Methodenentwicklung : Diskursive Definition heuristischer prozeduraler Prozessmodelle als Beitrag zur Bewältigung von informationeller Komplexität im Produktleben.
ISBN 978-3-86644-285-6

Band 1 – 2009
KRAPPE, HARDY
Erweiterte virtuelle Umgebungen zur interaktiven, immersiven Verwendung von Funktionsmodellen.
ISBN 978-3-86644-380-8

Band 2 – 2009
ROGALSKI, SVEN
Entwicklung einer Methodik zur Flexibilitätsbewertung von Produktionssystemen. Messung von Mengen-, Mix- und Erweiterungsflexibilität zur Bewältigung von Planungsunsicherheiten in der Produktion.
ISBN 978-3-86644-383-9

Band 3 – 2009
FORCHERT, THOMAS M.
Prüfplanung. Ein neues Prozessmanagement für Fahrzeugprüfungen.
ISBN 978-3-86644-385-3

Band 1 – 2011
ERKAYHAN, ŞEREF
Ein Vorgehensmodell zur automatischen Kopplung von Services am Beispiel der Integration von Standardsoftwaresystemen.
ISBN 978-3-86644-697-7

Band 2 – 2011
MEIER, GUNTER
Prozessintegration des Target Costings in der Fertigungsindustrie am Beispiel Sondermaschinenbau.
ISBN 978-3-86644-679-3

Band 1 – 2012
Nicht erschienen

Band 2 – 2012
WUTTKE, FABIAN
Robuste Auslegung von Mehrkörpersystemen. Frühzeitige Robustheitsoptimierung von Fahrzeugmodulen im Kontext modulbasierter Entwicklungsprozesse.
ISBN 978-3-86644-896-4

Band 3 – 2012
KATIČIĆ, JURICA
Methodik für Erfassung und Bewertung von emotionalem Kundenfeedback für variantenreiche virtuelle Produkte in immersiver Umgebung.
ISBN 978-3-86644-930-5

Die Bände sind unter www.ksp.kit.edu als PDF frei verfügbar oder als Druckausgabe bestellbar.

Band 1 – 2013	LOOS, MANUEL NORBERT Daten- und termingesteuerte Entscheidungsmethodik der Fabrikplanung unter Berücksichtigung der Produktentstehung. ISBN 978-3-86644-963-3
Band 2 – 2013	SYAL, GAGAN CAE - PROCESS AND NETWORK: A methodology for continuous product validation process based on network of various digital simulation methods. ISBN 978-3-7315-0090-2
Band 1 – 2016	BURGER, ALEXANDER Design for Customer: Methodik für nachhaltige Kundenlösungen unter Zuhilfenahme eines bedürfnisorientierten Leistungskonfigurators. ISBN 978-3-7315-0168-8
Band 2 – 2016	HOPF, JENS MICHAEL Framework for the Integration of Mobile Device Features in PLM. ISBN 978-3-7315-0498-6
Band 1 – 2017	WALLA, WALDEMAR Standard- und Modulbasierte digitale Rohbauprozesskette: Frühzeitige Produktbeeinflussung bezüglich Produktionsanforderungen im Karosserierohbau der Automobilindustrie. ISBN 978-3-7315-0600-3
Band 1 – 2018	WEISER, ANN-KATRIN Methodik eines holistischen Variantenmanagements modularer Produktfamilien – Grundlagen, Systematik und beispielhafte Anwendung der VM_{ahead} Methodik. ISBN 978-3-7315-0775-8
Band 2 – 2018	STANEV, STILIAN Methodik zur produktionsorientierten Produktanalyse für die Wiederverwendung von Produktionssystemen – 2REUSE. Konzept, Informationsmodell und Validierung am besonderen Beispiel des Karosserierohbaus in der Automobilindustrie. ISBN 978-3-86644-932-9

Band 1 – 2019
KLINGER, JULIUS FRIEDRICH
Tolerance Simulation in the Loop: Ansätze zur Verbesserung der Vorhersagegenauigkeit der Toleranzsimulation im Automobilbau durch Adaption an reale Fertigungsprozesse.
ISBN 978-3-7315-0876-2

Band 2 – 2019
SALEHI, MEHDI
Bayesian-Based Predictive Analytics for Manufacturing Performance Metrics in the Era of Industry 4.0.
ISBN 978-3-7315-0908-0

Band 3 – 2019
ELSTERMANN, MATTHES
Executing Strategic Product Planning – A Subject-Oriented Analysis and New Referential Process Model for IT-Tool Support and Agile Execution of Strategic Product Planning.
ISBN 978-3-7315-0972-1

Band 1 – 2021
SCHNEIDER, MARCUS
Methodik für Wissens- und Prozessmanagement bei der interaktiven kollaborativen Montage variantenreicher Produkte.
ISBN 978-3-7315-1046-8

Band 1 – 2023
LOHSE, OLIVER
Entwicklung einer Methode zum Einsatz von Reinforcement Learning für die dynamische Fertigungsdurchlaufsteuerung.
ISBN 978-3-7315-1282-0

ISSN 1860-5990
ISBN 978-3-86644-1282-0

Gedruckt auf FSC-zertifiziertem Papier

